

ALGORITHMS FOR LINEAR AND CONVEX  
FEASIBILITY PROBLEMS:  
A BRIEF STUDY OF ITERATIVE PROJECTION,  
LOCALIZATION  
AND SUBGRADIENT METHODS

A DISSERTATION  
SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING  
AND THE INSTITUTE OF ENGINEERING AND SCIENCE  
OF BILKENT UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

By  
Hakan Üzaktay  
August 1998

THESIS  
T  
57.74  
.093  
1998



ALGORITHMS FOR LINEAR AND CONVEX  
FEASIBILITY PROBLEMS:  
A BRIEF STUDY OF ITERATIVE PROJECTION,  
LOCALIZATION  
AND SUBGRADIENT METHODS

A DISSERTATION  
SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING  
AND THE INSTITUTE OF ENGINEERING AND SCIENCES  
OF BILKENT UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

By  
Hakan Özaktaş  
August 1998

T

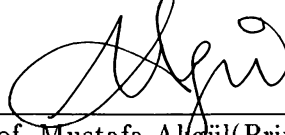
57.74

-093

1998

*B*043933

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



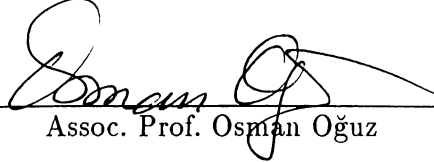
Assoc. Prof. Mustafa Akgül(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



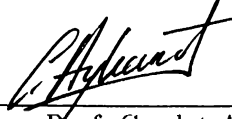
Assoc. Prof. Mustafa Ç. Pinar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



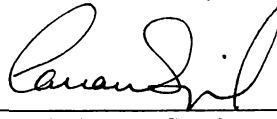
Assoc. Prof. Osman Oğuz

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



Assoc. Prof. Cevdet Aykanat

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



Assoc. Prof. Canan Sepil

Approved for the Institute of Engineering and Sciences:



Prof. Mehmet Baray  
Director of Institute of Engineering and Sciences

## ABSTRACT

### ALGORITHMS FOR LINEAR AND CONVEX FEASIBILITY PROBLEMS: A BRIEF STUDY OF ITERATIVE PROJECTION, LOCALIZATION AND SUBGRADIENT METHODS

Hakan Özaktaş

Ph.D. in Industrial Engineering

Supervisor: Assoc. Prof. Mustafa Akgül

August 1998

Several algorithms for the feasibility problem are investigated. For linear systems, a number of different block projections approaches have been implemented and compared. The parallel algorithm of Yang and Murty is observed to be much slower than its sequential counterpart. Modification of the step size has allowed us to obtain a much better algorithm, exhibiting considerable speedup when compared to the sequential algorithm. For the convex feasibility problem an approach combining rectangular cutting planes and subgradients is developed. Theoretical convergence results are established for both cases. Two broad classes of image recovery problems are formulated as linear feasibility problems and successfully solved with the algorithms developed.

*Key words.* Linear feasibility, convex feasibility, projection methods, the relaxation (successive orthogonal projections) method, Cimmino's method, surrogate constraints and block projections, long-step methods, sequential and parallel algorithms, subgradient methods, central cutting (localization) methods, analytic centers, descent directions, image recovery, image restoration, image reconstruction from projections, tomography, regularization of ill conditioned problems.

## ÖZET

### LİNEER VE KONVEKS FİZİBİLİTE PROBLEMLERİ İÇİN ALGORİTMALAR

Hakan Özaktaş

Endüstri Mühendisliği Bölümü Doktora

Tez Yöneticisi: Doç. Dr. Mustafa Akgül

Ağustos 1998

Bu çalışmada fizibilite problemi için çeşitli algoritmalar incelenmektedir. Lineer sistemlerde birkaç blok projeksiyon yaklaşımı uygulanmış ve kıyaslanmıştır. Yang ve Murty'nin paralel algoritmasının dizisel yaklaşımlardan çok daha yavaş olduğu gözlenmiştir. Adım boyunun düzeltilmesi sonucu dizisel algoritmalarından daha hızlı bir paralel algoritma elde edildiği görülmüştür. Konveks fizibilite problemine ise dik kesmeli ve alttürevsel yöntemleri birleştiren bir yaklaşım getirilmiştir. Her iki durum için de teorik sonuçlar verilmiştir. Fizibilite probleminin görüntü düzeltmedeki uygulamalarına dikkat çekilmiş, incelenen iki değişik problem için başarılı sonuçlar alınmıştır.

*Anahtar sözcükler.* Lineer fizibilite, konveks fizibilite, projeksiyon yöntemleri, Kaczmarz yöntemi, Cimmino yöntemi, aracı kısıtlar ve blok projeksiyonlar, uzun adımlı yöntemler, dizisel ve paralel algoritmalar, alttürevsel yöntemler, merkezi kesme (lokalizasyon) yöntemleri, analitik merkezler, yokuş yönleri, görüntü düzeltme, görüntü restorasyonu, görüntü rekonstrüksiyonu, tomografi, kötü davranımlı problemlerin regülasyonu.

“... Science tells us what we can know. but what we can know is little, and if we forget how much we cannot know we become insensitive to many things of very great importance. Theology, on the other hand, induces a dogmatic belief that we have knowledge where in fact we have ignorance, and by doing so generates a kind of impertinent insolence towards the universe. Uncertainty, in the presence of vivid hopes and fears is painful, but must be endured if we wish to live without the support of comforting fairy tales.”<sup>1</sup>

---

<sup>1</sup>Bertrand Russell, *History of Western Philosophy*, 1946.

## ACKNOWLEDGMENTS

I am grateful to Mustafa Akgül for his interest in my research and for supervising me with patience throughout my graduate and undergraduate studies. Without his help and guidance it would be quite impossible for me to put my work into written material.

I am indebted to Mustafa Pınar for his everlasting interest in my studies. His comments, suggestions and help were indispensable for the accomplishment of this work.

I am also grateful to Cevdet Aykanat for his support during parallel implementation of the algorithms. More than that, his comments and advice have been most beneficial.

I am grateful to Osman Oğuz for his encouragement, as well as his willingness to act as a committee member. I am also grateful to Canan Sepil for her comments on my work as a committee member.

My sincere gratitude should go to Tahsin Kurç for his generous support during compilation of the parallel algorithms with PVM. Were it not for his help, it would be impossible for me to obtain the parallel timing results presented. I am also grateful to Susanne Stassen for showing interest in my work and her efforts to obtain better timing results. I am grateful to Fatih Erden, Alper Kutay and Haldun Özaktaş for helping me with the formulation of image restoration problems. I am also indebted to Ergin Atalar for discussions on image recovery and reconstruction.

At this point I would also like to express my sincere thanks to the Commission of the European Communities, Directorate General for Industry (under contract ITDC



204-82166) and the Scientific and Technical Research Council of Turkey, TÜBİTAK (under grant EEEAG 160) for partially supporting this research.

It would be quite impossible for me to overcome my computer troubles all alone. I am grateful to the BCC staff, my office-mates and all, who passed by my office to whom I have asked quite frequently for assistance. In particular I would like to thank Erkan Uçar for aiding me with the difficulties I have had with *UNIX*. I am also grateful to Ali Gündüz and Gökhan Moral for installing *LINUX* on my PC which saved me from *Windows 95*.

I would also like to express my gratitude to Yair Censor and Krzysztof Kiwiol for their interest in my work. Their comments and criticisms have been most invaluable.

I am also grateful to the staff of Bilkent University. In particular, I am especially thankful to Fulya Çakırlar, Yeşim Karadeniz, Bilge Aydın, Gülseren Oskay, Arzu Açıkbaş, Tülin Oğuz, Öznur Koç, Zekeriya Kaya and İmral Saka for their kind help during my studies.

I am indebted to our faculty members for their support during my undergraduate and graduate years. In particular I am grateful to Ömer Benli, Barbaros Tansel, Cemal Dinçer, Halim Doğrusöz and Béla Vizvári. My sincere thanks should also go to Selim Aktürk, Ülkü Gürler, Vladimir Anisimov, İhsan Sabuncuoğlu and Oya Karaşan.

I also wish to express my gratitude to all graduate students and friends in the department. I am most thankful to Muhittin Demir, Bahar Kara Yetiş and Alev Kaya who helped me in any and every way. I am grateful to Barış Balcıoğlu who has always been praising my work without really knowing the subject matter. My sincere thanks should also go to my office-mates, in retrospect: Beşir Amcaoğlu, Nebahat Dönmez, Sıtkı Timuçin, Selçuk Avcı, Alper Atamtürk, Elif Görgülü, Vedat Verter, Mohamud Mohamed, Ceyda Oğuz, Cemal Akyel and Esra Doğan, with whom I have had memorable times. I am also grateful to my neighboring office-mates Deniz Özdemir, Ayten Türkcan, Evrim Güneş, Hande Yaman, Gürhan Kök, Bahar Deler, Eylem Tekin, Ayşegül Toptal, Maher Lahmar, Tahar Lejmi, Pelin Arun, Serkan Özkan and Ali Erkan. Also again in retrospect, I would like to remember Aslı Sencer, İhsan Durusoy, Hakan Demirel, Burçkaan Gürgün, Süleyman Karabük, Nureddin Kırkavak, Levent Kandiller, Oktay Yırtıcı, Fatih Yılmaz, Yavuz

Günelay, İradj Ouveysi, Noyan Narin, Ediz Urhan, Hasan Bala, Sıla Çetinkaya, Mine Alp Çağlar, Haluk Yılmaz, Okan Balköse, Orhan Dağhoğlugil, Dilek Yılmaz, Aslıhan Tabanoğlu, Sibel Salman, Erhan Kutanoğlu, Burçin Bozkaya, Aydın Selçuk, Gülcan Yeşilkökçen, Pınar Keskinocak, Sertaç Köksaldı, Mehmet Özkan, Özgür Atilla Tüfekçi, Murat Bayız, Abdullah Daşcı, Alper Şen, Engin Topaloğlu, Abdullah Çömlekçi, Savaş Dayanık, Samir Elhedhli, Hülya Emir, Erdem Gündüz, Feryal Erhun, Fatma Gzara, Mustafa Karakul, Chokri Hamdaoui, Kemal Kılıç, Muzaffer Tanyer, Bayram Yıldırım and Murat Aksu.

Finally, I have to express my sincere thanks to the many people who have been of help, but which I have forgotten to mention here.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Classification of the Approaches in the Literature</b>	<b>3</b>
2.1	Central Cutting Methods . . . . .	3
2.2	Subgradient Direction Methods . . . . .	4
2.3	Projection Methods	5
<b>3</b>	<b>The Linear Feasibility Problem</b>	<b>6</b>
3.1	On Orthogonal Projections onto Convex Sets . . . . .	7
3.2	The Sequential Yang-Murty Algorithm . . . . .	8
3.3	The Parallel Yang-Murty Algorithm . . . . .	10
3.4	Ideas for Improving Parallel Performance . . . . .	13
3.5	A Longer Step Size . . . . .	16
3.6	Implementation Results for Random Problems . . . . .	22
3.7	Notes on Implementation . . . . .	26
3.7.1	Implementation Data and Test Parameters . . . . .	26

<i>CONTENTS</i>	x
3.7.2 Parallel Architecture . . . . .	28
3.8 Generalization of the New Step Size to Convex Feasibility Problems and Subgradient Methods . . . . .	30
<b>4 The Convex Feasibility Problem</b>	<b>31</b>
4.1 A Variety of Centers . . . . .	32
4.2 The Ye-Goffin Approach of Analytic Centers	33
4.3 Multiple Cuts . . . . .	37
4.4 Estimation of the Shrinkage Rate of Containing Domain Procedures	41
4.5 An Approach Combining Containing Domains and Descent Directions	42
4.6 Obtaining Rectangular Cuts Analytically and the Primitive Rectan- gular Cutting Plane Algorithm . . . . .	43
4.7 A New Algorithm Based on Containing Boxes and Subgradients . . .	47
4.8 Some Concepts Related to the Descent Approach . . . . .	50
4.9 Convergence Results . . . . .	56
4.9.1 Some Lemmas Related to Convergence . . . . .	57
4.9.2 Establishing Convergence of the Extended Version of the Rectangular Cutting Plane Approach . . . . .	61
<b>5 Applications</b>	<b>68</b>
5.1 Image Recovery Formulated as a Linear Feasibility Problem . . . . .	70
5.2 Pre-filtering and Smoothing . . . . .	71
5.3 Restoration of Space Variant Global Blurs Caused by Severe Camera Movements and Coordinate Distortions . . . . .	73

<i>CONTENTS</i>	xi
5.4 Image Recovery in the Presence of Severely Space Variant Geometric Distortions and Point Spread Functions	78
5.5 Notes on Implementation of the Block Projections Algorithm in Image Restoration Applications . . . . .	81
<b>6 Conclusions and Prospects for the Future</b>	<b>83</b>
<b>Vita</b>	<b>94</b>



# Chapter 1

## Introduction

In this research we present a study of iterative procedures for feasibility problems. We have analyzed these problems in two groups. The first group consists of linear feasibility problems. In this group, we have considered very large scaled, sparse and unstructured systems where traditional pivoting or elimination techniques and decomposition methods are considered to be inefficient. The second group consists of nonlinear convex feasibility problems. In our studies we have mostly assumed an explicit definition of a convex set  $\Gamma$  of the following form:

$$\Gamma = \{y \in \mathfrak{R}^m : f_i(y) \leq 0, \quad i = 1, 2, \dots\} \quad (1.1)$$

where  $\forall f_i$  are convex functions in  $\mathfrak{R}^m$ .

For linear problems we have concentrated on block projections methods. Block projections methods for the linear feasibility problem make use of projections onto a surrogate constraint representing a set of violated constraints, and are much more efficient than algorithms which make projections onto distinct constraints.

We have coded and tested the sequential and parallel versions of the block projections algorithm of Yang and Murty. It has turned out that the performance of the parallel version is significantly inferior to the sequential version. Our main contribution to this problem has been to obtain a parallel algorithm which performs better than the sequential algorithm, through a modification of the step sizing rule. This modification has been justified both theoretically and practically. Furthermore, the modified step sizing rule can be extended to parallel projection algorithms for

the general convex feasibility problem as well. In addition to random test problems we have also tested our procedure on real life applications in image restoration.

For the convex feasibility case, we have first considered central cutting plane approaches, particularly the algorithms of Ye and Goffin. After examining the use of multiple cuts and descent directions instead of center calculations in these approaches, we have introduced the concept of rectangular cuts to obtain an algorithm which maintains a simple containing domain. The use of descent directions has served the aim of obtaining better iterates than central points. Following these two ideas we have constructed a new algorithm, and verified that this algorithm uses a descent direction with respect to a reference function representing the convex feasibility problem. The algorithm produces a monotonically decreasing sequence of the reference function values (and is convergent) when a line search subroutine is called by the main algorithm. Furthermore, the algorithm still converges when a self adaptive step sizing subroutine which can provide an overall descent of the functional values of the generated sequence is used.

In the next chapter we will present a general overview of the approaches in the literature for the feasibility problem. In the third chapter we describe our developments (both theoretical and practical) for the block projections methods. In the fourth chapter we present our work on the convex feasibility problem. In the fifth chapter we introduce two novel applications of the linear feasibility problem in image processing.

Within each chapter we have tried to keep our notation consistent. However, there are some differences between chapters. It was our objective to be in accordance with the notation of the major references. Thus in Chapter 3 the problem is to determine  $x \in K \subseteq \mathfrak{R}^n$ ,  $K$  being a nonempty polyhedron [Yang & Murty 92]; in Chapter 4 the problem is to find  $y \in \text{int}(\Gamma) \subseteq \mathfrak{R}^m$ ,  $\Gamma$  being a convex set with nonempty interior [Gof. *et al.* 93a, b], [Luo & Sun 95], whereas in Chapter 5 we represent the linear feasibility problem as  $A\xi \leq b$  [Özak. *et al.* 98a, b].

## Chapter 2

# Classification of the Approaches in the Literature

This chapter covers several iterative routines which have been devised to solve the convex feasibility problem. We have classified these routines into three categories: (i) central cutting methods, (ii) subgradient direction methods, and (iii) projection methods. We should note, however, that these categories are not completely disjoint.

### 2.1 Central Cutting Methods

The central cutting methods (which are also referred to as localization methods) are ‘test and cut’ routines. The feasible region is assumed to be a subset of a containing domain and at each iteration a point within this domain is computed as the new iterate. This point is tested for feasibility and if this is not the case, the containing domain is shrunk somehow (for example by several valid cuts through the current point) and a new test point is regenerated. The routine stops when a feasible (interior) point is found.

Convergence of these algorithms is closely related to the shrinkage rate of the containing domain. So the selection of the iteration point within the containing domain is an important issue. As one gets closer to a center point of this domain, it becomes more likely to obtain deeper cuts so that shrinkage will be faster

[Nemirovsky & Yudin 83]. For this reason, a subroutine for computing an almost exact or an approximate center of the current domain is required.

The ellipsoid method [Khachiyan 79] is an example for these algorithms (with some differences of the general framework given above, though). The containing domains are ellipsoids which are generated successively and the test points are the centers of these ellipsoids. The convex optimization algorithm of volumetric centers given in [Vaidya 89], employs containing polytopes which contract iteratively by the introduction of new inequalities. The algorithms given in [Ye 89], [Gof. *et al.* 93a, b] also utilize polytopes as containing domains and the so called analytic centers (of polytopes) which are relatively easier to compute [Gritz. & Klee 93a]. The algorithm outlined in [Luo & Sun 95] solves a convex quadratic feasibility problem by shrinking convex bodies and testing at analytic centers, redefined for convex sets. Some extensions to general convex optimization are given in [Altman & Kiwiel 96].

## 2.2 Subgradient Direction Methods

To find a feasible point satisfying the inequalities  $f_i(y) \leq 0$ ,  $i = 1, \dots, p$  by an iterative routine, a natural way is to use the negated subgradients as the movement direction. Such algorithms are usually referred to as descent methods [Censor & Lent 82], [Kiwiel 85], [Schein. & Oliv. 92], [Kiwiel 96a, b]. For convenience we have assumed differentiability of these functions throughout the feasible domain of  $y$  (however, our results can be generalized to the nondifferentiable case simply by replacing gradients by subgradients, without any further assumption). Having made this assumption, we can regard these movement directions as steepest descent directions for individual functions. Instead of reaching a minimum of a single function, one tries to reduce several functions below the limiting value of 0.

In Chapter 4, movements in the reverse directions of gradients will be combined with the containing domain approaches. The motivating idea behind these search directions is to obtain better test points than the center points. In this way one can expect faster shrinkage of the containing domain and faster convergence of the generated sequence to the feasible region.

## 2.3 Projection Methods

The orthogonal projection of a point  $\bar{x}$  onto a convex set  $C$  is a point (if it exists) of the convex set which has the minimal Euclidean distance to  $\bar{x}$ . Nonemptiness is required by assumption, however existence of an interior point (full dimensionality) is not necessarily the case.

Projection methods date back to the early works of Kaczmarz and Cimmino in the thirties. Both approaches are iterative procedures which solve systems of linear equations. In the Kaczmarz approach, projections are made onto hyperplanes (which represent linear equations) successively, whereas Cimmino's method is based on simultaneous projections onto all, and takes their convex combination.

The same idea can be applied to the solution of a set of linear inequalities. The successive projections approach of Kaczmarz (which is also referred to as the relaxation method), has been generalized to the case of inequalities. Convergence results for this method as given in [Agmon 54], [Motzkin & Scho. 54] are quite fundamental and they still serve as the basis for various projection approaches.

These approaches have become popular starting from the seventies with the development of the algebraic reconstruction technique (ART) for computerized tomography.

During the implementation of these methods, one has to specify a certain feasibility tolerance (to enable finite convergence), since projection algorithms tend to converge infinitely to the boundary of the feasible region.

Computing the projection onto an arbitrary convex set is a nontrivial problem. However, for the case of a linear equation (or inequality) it is quite straightforward. Actually, the projection routine for linear systems is a special case of the subgradient methods where the step sizes are selected with respect to their distances to the hyperplanes (hence distances to the related half spaces). Just like the subgradient methods, these routines are nonpolynomial time algorithms [Goffin 80], almost all with linear rates of convergence. They are practical for very large linear problems with sparse and unstructured coefficient matrices.



## Chapter 3

# The Linear Feasibility Problem

The linear feasibility problem, though it might seem to be trivial at a first glance, is quite challenging when the matrix dimensions are large. The fundamental Fourier-Motzkin elimination technique (see for example [Dantzig & Eaves 73]) is not realistic to implement for many real world problems [Chvátal 83]. Other direct (noniterative) methods such as LP pivoting or Gaussian elimination may also be inefficient when the underlying matrix is huge and sparse with an irregular nonzero pattern.

Nice reviews of the projection methods for the feasibility problem are given in [Combettes 93] and [Censor & Zenios 97]. In a recent paper, a generalized framework for the projection approaches in the literature has been given [Bauschke & Borwein 96].

In this chapter we present an analysis of the block projections methods outlined in [Yang & Murty 92]. Similar routines have been given in [Censor 88], [Aharoni & Censor 89], [Gar.-Pal. 90], [Oko 92], [Gar.-Pal. 93], [Kiwiel 95], [Gar.-Pal. & Gon.-Cas. 96] as well. Our point of departure was to compare the sequential and parallel versions of the Yang-Murty algorithm. We have observed that the parallel block approach of Yang-Murty (as well as many similar routines in the literature) perform quite poorly in practice. Instead of the conventional short steps used in the simultaneous methods we have considered longer steps based on ideas outlined in [Kiwiel 95] and [Gar.-Pal. & Gon.-Cas. 96]. The resulting performance with this step size modification is superior to that obtained with the sequential algorithm. The results of this research have been reported in [Özak. *et al.* 96, 97a, 97b].

In the next section we present some basic knowledge related to the projection methods. In Sections 3.2 and 3.3 we outline the sequential and parallel versions of the Yang-Murty algorithm which we have implemented. In Section 3.4 we suggest a new step size to improve the performance of the parallel routine. In Section 3.5 we present an analytical verification of the convergence of the parallel algorithm with the new step size. In Sections 3.6 and 3.7 we compare implementation results for the sequential and several parallel routines. In Section 3.8 we generalize the convergence results to the convex feasibility problem.

### 3.1 On Orthogonal Projections onto Convex Sets

A projection of a point  $x^k \in \mathfrak{R}^n$  onto a convex set  $C_i$  gives the point (if there is any)  $x \in C_i$  which has the minimal Euclidean distance to  $x^k$  [Hir.-Urr. & Lemar. 93]. More generally, projections are defined as the nearest points contained in the convex bodies with respect to appropriate distance definitions.

Usually, one has to compute the projection of  $x^k$  onto  $C_i$  when  $x^k \notin C_i$ . This projection requires the solution of the following problem:

$$P_{C_i}(x^k) = \min_{x \in C_i} \|x^k - x\| \quad (3.1)$$

In this case, the minimization is made over the Euclidean distance, and the nearest point of  $C_i$  to  $x^k$  is found. Note that, existence of this minimum does not imply that the projection is an orthogonal projection.

For the cases where  $C_i = \{x \in \mathfrak{R}^n : f_i(x) \leq 0\}$ ,  $f_i(x)$  convex and differentiable, the direction  $(P_{C_i}(x^k) - x^k)$  is given by  $\nabla f_i(x^{k+1})$ . However, to determine this direction and the next point, one needs to solve the minimization problem defined above and in a way, to find the point  $x^{k+1}$  in advance. We will state two propositions (with elementary verifications) on the existence of orthogonal projections. As stated above, an orthogonal projection may not exist for certain cases. We will assume a finite dimensional Euclidean space,  $\mathfrak{R}^n$ .

**Proposition 1.** Let  $C_i$  be a closed convex set with nonempty interior and  $\bar{x}$  be a point such that  $\bar{x} \notin C_i$ . Then an orthogonal projection of  $\bar{x}$  onto  $C_i$  does exist.

**Proposition 2.** Let  $C_i$  be a subspace (or a linear variety) of  $\mathfrak{R}^n$  and  $\bar{x}$  be a

point such that  $\bar{x} \notin C_i$ . Then an orthogonal projection of  $\bar{x}$  onto  $C_i$  does exist.

From **Proposition 2** one can say that a projection onto a linear equation (hence to a linear inequality induced by this equation) exists. In our subsequent discussions, it will be assumed for convenience that an orthogonal projection exists.

The convergence of the projection methods are based on the nonexpansivity property which is related to the Fejér-monotonicity of the sequence generated by the algorithm. These important definitions will be stated below (see for example [Crombez 91]).

**Definition 1.** Let  $\bar{x} \in \mathfrak{R}^n$ ,  $C_i \subseteq \mathfrak{R}^n$  and  $P$  be a mapping  $\mathfrak{R}^n \rightarrow \mathfrak{R}^n$ .  $P$  is referred to as a nonexpansive mapping, if for every  $f \in C_i$  it satisfies:

$$\|\bar{x} - f\| \leq \|P\bar{x} - f\| \quad (3.2)$$

**Definition 2.** Let  $C_i \subseteq \mathfrak{R}^n$ ,  $P$  a nonexpansive mapping and let  $\{x^n\}$  be a sequence generated by  $P$ .  $P$  is said to be Fejér-monotone with respect to  $C_i$  if for every  $f \in C_i$  and  $k$  it satisfies:

$$\|x^{k+1} - f\| \leq \|x^k - f\| \quad (3.3)$$

A detailed treatment of the concepts given above can be found in [Bauschke & Borwein 96]. It has to be stated that Fejér-monotonicity can be satisfied by arbitrary projections onto separating hyperplanes (instead of the set  $C_i$  itself) so that one can refrain from computing complicated projections, and this idea forms the basis of the  $(\delta, \eta)$  approach [Censor & Zenios 97].

## 3.2 The Sequential Yang-Murty Algorithm

The problem which will be of main interest to us in this chapter is of the type:

$$Ax \leq b \quad (3.4)$$

where  $x \in \mathfrak{R}^n$  and  $A$  is  $m \times n$ . By assumption, the feasible set  $K$  defined by this inequality is nonempty.

The method of successive orthogonal projections (SOP) of Gubin, Polyak and Raik works by projecting the current point onto a convex set at each iteration until a point which is contained in the intersection of these convex sets is found. At a typical iteration of the SOP algorithm (actually in many projection algorithms) an overrelaxed or an underrelaxed step is taken in the computed projection direction. Hence an iterative step becomes:

$$x^{k+1} = x^k + \lambda_k(P_{C_i}(x^k) - x^k) \quad 0 < \lambda_k < 2 \quad (3.5)$$

where  $P_{C_i}$  is the projection operator onto the closed convex set  $C_i$  and  $\lambda_k$  is the so called relaxation parameter. When  $\lambda_k = 1$ , the next point generated is the exact orthogonal projection of the current point. On the other hand, when  $\lambda_k > 1$ , one has a longer step, which is the case of overrelaxation and when  $\lambda_k < 1$ , one has a shorter step, which is the case of underrelaxation [Censor & Zenios 97].

In an explicitly defined linear feasibility problem the intersection of many halfspaces define the convex set. At an iteration point  $x^k$ , the routine proceeds by considering a violated inequality  $A_i x^k > b_i$  and calculating the next point as:

$$x^{k+1} = x^k - \lambda_k \frac{A_i x^k - b_i}{\|A_i\|^2} A_i \quad (3.6)$$

where  $0 < \lambda_k < 2$ .

The relaxation method is quite inefficient when the number of inequalities is large [Yang & Murty 92], [Gar.-Pal. 93]. Considering the projection onto a surrogate plane (instead of distinct constraining planes) is a useful idea. This is simply achieved by defining a hyperplane  $\pi^k(Ax - b) = 0$  where the  $i^{\text{th}}$  component of the row vector  $\pi^k$  is positive if the current test point  $x^k$  does not satisfy (3.4).

Since the number of constraints in (3.4) is quite large, partitioning the matrix into blocks and making surrogation within these blocks is easier to implement. Additionally, partitioning the matrix (rowwise) increases the efficiency of the projection algorithm to some extent.

As is the case in many applications such as those arising in image reconstruction and restoration it is assumed that the matrix  $A$  is sparse and unstructured, so it is logical to partition it into equal (or almost equal) blocks. Let each block  $t = 1, \dots, p$  consist of  $m_t$  rows so that each partition may be denoted as  $A^t x \leq b^t$ . The surrogate constraint is defined as  $\pi^t A^t x \leq \pi^t b^t$  (which is clearly a valid inequality), where  $\pi^t$

is a nonnegative weight vector. The following algorithm, which is referred as the ‘sequential surrogate constraint method’, is given in [Yang & Murty 92]:

**The Sequential Surrogate Constraint Algorithm:**

**Step 0.** Generate or read a feasible problem, with  $A \in \Re^{m \times n}$ ,  $b \in \Re^m$  with previously known values of  $n$ ,  $m$ ,  $p$ ,  $m_1, \dots, m_p$ . Initially, let  $k = 0$  and  $t = 1$ . Fix a value of  $\lambda$  so that  $0 < \lambda < 2$ .

**Step 1.** Check if  $A^t x^k \leq b^t$ . If so, then let  $x^{k+1} = x^k$ . Otherwise let

$$x^{k+1} = x^k - \lambda \frac{(\pi^{t,k} A^t x^k - \pi^{t,k} b^t)(\pi^{t,k} A^t)}{\|\pi^{t,k} A^t\|^2} \quad (3.7)$$

where  $\pi_i^{t,k} > 0$  if constraint  $i$  is violated, and  $\pi_i^{t,k} = 0$  otherwise. ( $\sum_{i=1}^{m_t} \pi_i^{t,k} = 1$  is required for convenience.) Update the value of the counter of violated inequalities.

**Step 2.** If  $t < p$ , let  $k = k + 1$ ,  $t = t + 1$  and go to **Step 1**. If  $t = p$  and if the total number of violated constraints in the major iteration is zero then stop, the current solution is feasible. Otherwise, assign zero as the new value of the counter of violated constraints, let  $t = 1$ ,  $k = k + 1$  and go to **Step 1**.

Verification of convergence is based on the Fejér-monotonicity of the generated sequence  $\{x^k\}_{k=0}^{\infty}$ . If the feasibility check in **Step 1** is adjusted to allow for a certain degree of tolerance so that  $A_i x^k$  is compared with  $b_i + \varepsilon$ , then the algorithm converges finitely [Yang & Murty 92].

### 3.3 The Parallel Yang-Murty Algorithm

Cimmino’s method is an iterative routine which makes simultaneous projections onto all violated constraints and takes their convex combination as the iterative step. However, it is impossible to implement when the number of constraints is large (even more severely than the relaxation method).

The simultaneous block projections approach is the parallel version of the method described in the previous section. In the simultaneous case all submatrices are processed on distinct machines at the same time. Instead of adding  $p$  successive



projections on top of each other, their convex combination is computed, and a single combined step is taken at a major iteration.

The modified algorithm has been outlined in [Yang & Murty 92]. Theoretical convergence results are quite similar to those of the sequential case. It should be stated that the block projections approach is not unique in the literature. A generalized approach of the Yang-Murty algorithm (both parallel and sequential versions) has been described earlier in [Censor 88] and [Aharoni & Censor 89], and also in [Kiwiel 95]. Other surrogate approaches include [Dos Santos 87], [Oko 92], [Gar.-Pal. 93] and [Gar.-Pal. & Gon.-Cas. 96].

The parallel algorithm is given as follows. We consider equal weights ( $\tau_t$ 's) for equal sized partitions (similar to the sequential case) since none of the blocks have a structural superiority to others.

### The Parallel Surrogate Constraint Algorithm:

**Step 0.** Generate or read a feasible problem. Let  $k = 0$  and fix  $\lambda$  so that  $0 < \lambda < 2$ .

**Step 1.** For  $t = 1, \dots, p$   
check if  $A^t x^k \leq b^t$ . If so, then let  $P_t(x^k) = x^k$ . Otherwise let

$$P_t(x^k) = x^k - \frac{(\pi^{t,k} A^t x^k - \pi^{t,k} b^t)(\pi^{t,k} A^t)}{\|\pi^{t,k} A^t\|^2} \quad (3.8)$$

where  $\pi^{t,k}$  is the same as in the sequential algorithm. When the entire matrix is processed, let  $P(x^k) = \sum_{t=1}^p \tau_t P_t(x^k)$  where  $\sum_{t=1}^p \tau_t = 1$ ,  $\tau_t \geq 0$ , and  $\tau_t > 0$  for all blocks which violate feasibility. The next point is generated as:

$$x^{k+1} = x^k + \lambda(P(x^k) - x^k) \quad (3.9)$$

Update the total number of violated constraints, in all blocks.

**Step 2.** If the total number of violated constraints in the major iteration is zero then stop, the current solution is feasible. Otherwise, assign zero to the number of violated constraints, let  $k = k + 1$  and go to **Step 1**.

We can rewrite (3.9) as:

$$x^{k+1} = x^k - \lambda \left( \sum_{t=1}^p \tau_t d_t^k \right) \quad (3.10)$$

where,

$$d_t^k = \frac{(\pi^{t,k} A^t x^k - \pi^{t,k} b^t)(\pi^{t,k} A^t)}{\|\pi^{t,k} A^t\|^2} \quad (3.11)$$

As just stated, a unified framework for the block projections algorithms has been considered in the literature. Whether the algorithm is successive or simultaneous depends on the choice of the parameters. Hence we prefer to state a general form of the block projections algorithm in the following way:

### The Block Projections Algorithm:

**Step 0.** Let  $A \in \Re^{m \times n}$  and  $b \in \Re^m$ . Consider the even partitioning of the  $A$  matrix and the  $b$  vector into  $p$  rowwise blocks of almost equal sizes, as  $[A^1|A^2|\dots|A^p]^T$  and  $[b^1|b^2|\dots|b^p]^T$ . (In the case of parallel implementation  $p$  is equal to the number of processors.) Let  $x^0 = 0$  and  $k = 0$ .

**Step 1.** For  $t = 1, \dots, p$ , based on a selection of  $\tau_t$  values (satisfying  $\forall \tau_t \geq 0$  and  $\sum_{t=1}^p \tau_t = 1$ ) check if  $A^t x^k \leq b^t$  for all  $t$  with  $\tau_t > 0$ . If so, then let  $d_t^k = 0$ . Otherwise let

$$d_t^k = \frac{(\pi_t^k A^t x^k - \pi_t^k b^t)(\pi_t^k A^t)}{\|\pi_t^k A^t\|^2} \quad \text{and} \quad P_t(x^k) = x^k - d_t^k \quad (3.12)$$

where  $\pi_i^{t,k} > 0$  if constraint  $i$  is violated and  $\pi_i^{t,k} = 0$  otherwise. Define a weighted projection as:

$$P(x^k) = \sum_{t=1}^p \tau_t P_t(x^k) = \sum_{t=1}^p \tau_t (x^k - d_t^k) \quad (3.13)$$

where  $\sum_{t=1}^p \tau_t = 1$ ,  $\tau_t \geq 0$ . The next point is generated as:

$$x^{k+1} = x^k + \lambda_k (P(x^k) - x^k) \quad (3.14)$$

where  $0 < \lambda_k < 2$ . Notice that equivalently we can rewrite equation (3.14) as:

$$x^{k+1} = x^k - \lambda_k \left( \sum_{t=1}^p \tau_t d_t^k \right) \quad (3.15)$$

Update the total number of violated constraints in all blocks.

**Step 2.** If the total number of violated constraints in the iteration is zero then stop, the current solution is feasible. Otherwise, assign zero to the number of violated constraints, let  $k = k + 1$  and go to **Step 1**.

To guarantee convergence, for a given block index  $t$  for which  $A^t x^k \not\leq b^t$  for all  $k > k'$  (where  $k'$  is fixed to a sufficiently large value),  $\tau_t$  should take positive values for infinitely many  $k$  iterations (see [Aharoni & Censor 89] and [Kiwiel 95]). Verification of convergence is based on the Fejér-monotonicity of the generated sequence  $\{x^k\}_{k=0}^{\infty}$ . Note that this algorithm is more general and does not require the two following imposed conditions of the Yang-Murty algorithms: (i) that the relaxation parameter should be fixed to  $\lambda$ , (ii) that  $\tau_t > 0$  for all  $t$  with  $A^t x^k \not\leq b^t$ .

The given algorithm is referred to as the *successive* block projections algorithm (or is said to have *cyclic control*), and corresponds to the sequential surrogate constraint algorithm of Yang-Murty, if at each iteration only one block coefficient is nonzero ( $\tau_t = 1$  for a single  $t$ ) and at the following iteration only the block coefficient with the successive index is nonzero ( $\tau_{t+1 \pmod{p}} = 1$ ). On the other hand if one takes  $\tau_t > 0, \forall t$  such that  $A^t x^k \not\leq b^t$ , then it is referred as the *simultaneous* block projections algorithm, and corresponds to the parallel surrogate constraint algorithm of Yang-Murty. In the absence of additional information it is natural to take equal weights  $\tau_t$  for all blocks whose inequalities are not fully satisfied (since none of the blocks has priority with respect to the others) and zero weights for the blocks whose inequalities are fully satisfied. If the simultaneous algorithm is implemented on a parallel computer, then all submatrices are processed on distinct machines at the same time. Instead of proceeding with  $p$  successive projections, the convex combination of  $p$  simultaneous projections is computed, and a single combined step is taken at a major iteration.

### 3.4 Ideas for Improving Parallel Performance

A comparison of test results for the sequential and parallel algorithms (see Tables 3.1 and 3.2) reveals that the parallel version of the Yang-Murty algorithm performs much worse than the sequential version. The situation becomes even worse when the number of processors is increased.

Before concluding that simultaneous methods are not practical to implement, one should consider adjustment of the algorithm in order to benefit as much as possible from parallelization. A close examination of the parallel algorithm makes it apparent that the step taken in (3.9) is quite short when compared to the accumulated sequential steps in a major iteration of the first algorithm. This issue has also been discussed in [Gar.-Pal. 93]. Thus a remedy for the parallel algorithm should be able to compensate for this deficiency.

Let us recall that the combined movement direction induced by the infeasible blocks at the  $k^{\text{th}}$  iteration is:

$$\bar{d}^k = \sum_{t=1}^p \tau_t d_t^k \quad (3.16)$$

In the parallel algorithm this convex combination of the individual directions is taken. Assuming that  $d_t^k = 0$  when block  $t$  is feasible, by substitution and rearrangement (3.9) becomes (using  $\lambda_k$  instead of a fixed  $\lambda$ ):

$$x^{k+1} = x^k - \lambda_k \bar{d}^k \quad (3.17)$$

An apparently promising idea is to magnify this ‘averaged step’ in some proportion to make it comparable in norm to the accumulated steps of the sequential algorithm. An appropriate magnifying parameter is the number of violated blocks at the  $k^{\text{th}}$  iteration. The new iterate then becomes:

$$x^{k+1} = x^k - (\lambda_k \bar{p}^k) \bar{d}^k \quad (3.18)$$

where  $\bar{p}^k$  is the number of blocks which are infeasible at  $x^k$ . The idea is illustrated in Figure 3.1. The results with this step sizing policy is given in Table 3.3. Good

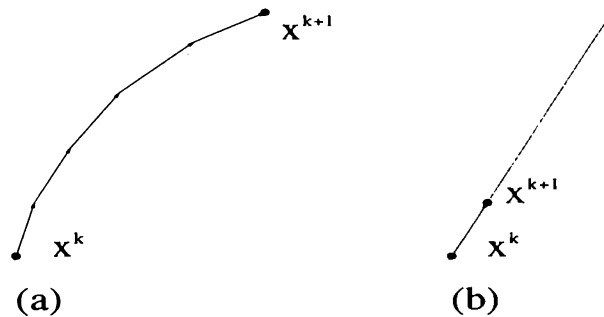


Figure 3.1: Movement in a major iteration in (a) sequential, and (b) parallel versions of the algorithm. The idea is to take a longer step in the same movement direction to achieve better convergence.

results are obtained for some problems, but unfortunately, the speed of convergence

becomes intolerable especially as the number of blocks are increased. Even worse, it has been observed that the algorithm gives divergent sequences for many examples. We can conclude that this idea may be useful if one can introduce some sort of a regulative mechanism which will prevent too long steps.

We have also tested the alternative step suggested recently in [Gar.-Pal. 93] and [Gar.-Pal. & Gon.-Cas. 96] for a similar simultaneous block projections algorithm. For the equal weighted case the suggested step is:

$$x^{k+1} = x^k - \frac{1}{2} \lambda_k \frac{\sum_{i=1}^p \|d_i^k\|^2}{\|\sum_{i=1}^p d_i^k\|^2} \sum_{i=1}^p d_i^k \quad (3.19)$$

It is shown both theoretically and practically in [Gar.-Pal. & Gon.-Cas. 96] that with this new step, the routine performs better (with respect to the parallel algorithm which uses the convex combination of surrogate steps) under some assumptions. Our results in Table 3.4 are also in accordance. However, the average number of major iterations are still unsatisfactory when compared to those of the sequential algorithm.

Returning to the iterative update given in equation (3.18), we consider the premultiplication of the step size with the fraction  $\frac{\sum \|d_i^k\|^2}{\|\sum d_i^k\|^2}$ , based on the *acceleration* idea discussed in [Gar.-Pal. & Gon.-Cas. 96]. One can notice after a closer examination of the behavior of the algorithm that, the generated sequence commences to diverge whenever this fraction is much smaller than unity. It is seen that as long as this fraction is around unity, the algorithm proceeds moderately and the sequence tends to converge. So one can vaguely conclude that when the fraction is relatively small, the current step is relatively long and has to be adjusted to enable convergence. So in addition to the magnifying parameter  $\bar{p}^k$ , we will propose to introduce the regulative parameter  $\frac{\sum \|d_i^k\|^2}{\|\sum d_i^k\|^2}$  in equation (3.18), so that the new iterate becomes:

$$x^{k+1} = x^k - \lambda_k \frac{\sum_{i=1}^p \|d_i^k\|^2}{\|\sum_{i=1}^p d_i^k\|^2} \bar{p}^k \bar{d}^k \quad (3.20)$$

Note that an equivalent statement of this new step can be given as:

$$x^{k+1} = x^k - \lambda_k \frac{\sum_{i=1}^p \|d_i^k\|^2}{\|\sum_{i=1}^p d_i^k\|^2} \sum_{i=1}^p d_i^k$$

Experimentation with this step sizing rule of the parallel algorithm has yielded encouraging results (Table 3.5). Furthermore, implementation of this algorithm on a parallel computer resulted in considerable speedups which were out of the question with the conventional short-step simultaneous algorithms. Convergence of



the modified simultaneous block projections algorithm obtained by replacing (3.9) (or identically (3.14)) with (3.20) will be given in the following section.

### 3.5 A Longer Step Size

Before establishing convergence, an intuitive explanation of the reasons behind the efficiency of the regulative parameter (motivated in the previous section) will be given. Considering two blocks, hence two movement directions  $-d_1, -d_2$ , this parameter is equivalent to  $\frac{\|d_1\|^2 + \|d_2\|^2}{\|d_1 + d_2\|^2} = \frac{\|d_1\|^2 + \|d_2\|^2}{\|d_1\|^2 + \|d_2\|^2 + 2\|d_1\|\|d_2\|\cos\theta}$  where  $\theta$  is the angle between  $-d_1$  and  $-d_2$ . Now when  $\theta$  is quite small (about 0–30 degrees), this ratio will be much less than 1 and premultiplication with this fraction will decrease the magnitude of the step originally given as  $-\lambda_k \bar{p}^k \bar{d}^k = -\lambda_k (d_1 + d_2)$ , substantially. When  $\theta$  is larger (about 70–90 degrees) this ratio will be above 1 and the magnitude will be reduced moderately. If however,  $\theta$  is larger than 90 degrees, the parameter will be above 1 and in this case the magnitude of the step will be increased. Thus, premultiplication with this regulative parameter works in both ways; it reduces steps which are too long and expands steps which are too short. It will be assumed that

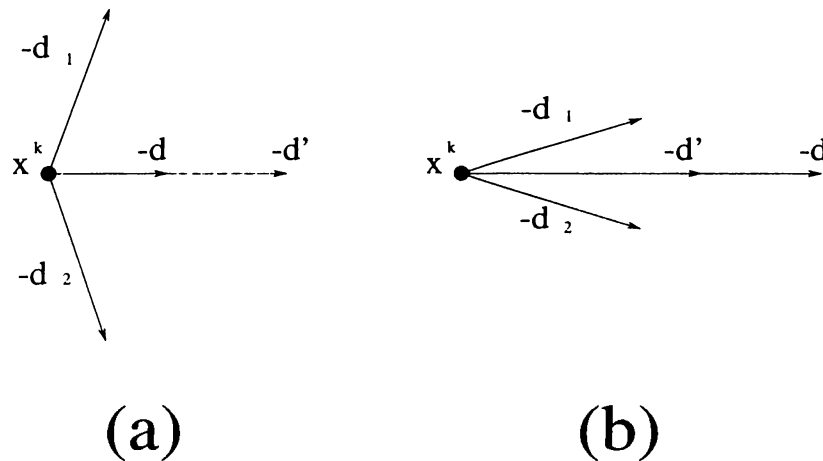


Figure 3.2: The parameter works in both ways. In (a)  $-d = -(d_1 + d_2)$  is a relatively short step, and multiplication with the regulative parameter results in  $-d'$ . In (b)  $-d = -(d_1 + d_2)$  is a relatively long step, and in this case multiplication with the regulative parameter we results in a moderate  $-d'$ .

the direction vectors,  $d_t : t = 1, \dots, p$  (we drop the iteration superscript  $k$  from  $d_t^k$  in this section) are linearly independent. Clearly, this assumption requires that  $p$  should be less than or equal to  $n$ , and that none of the surrogate plane pairs are

parallel. We define an 'overall surrogate plane' as:

$$\sum_{t=1}^p a_t x = \sum_{t=1}^p b'_t \quad (3.21)$$

where  $a_t = \pi^t A^t$  and  $b'_t = \pi^t b^t$ . Clearly, in place of equation (3.21) one can also write  $\sum_{t=1}^p d_t x = \sum_{t=1}^p b_t$  where  $b_t = b'_t (\|d_t\|/\|a_t\|)$ , for  $t = 1, \dots, p$ , with a little abuse of notation.

By definition, this plane contains the intersection set of the individual surrogate planes. The bunch of surrogate inequalities constitute the 'containing polyhedron', which clearly contains the set of feasible points  $K$ . (In the special case where  $p = n$  this polyhedron is a cone.) The overall surrogate plane supports the containing polyhedron so that the overall surrogate inequality  $\{x : \sum_{t=1}^p a_t x \leq \sum_{t=1}^p b'_t\}$  contains  $K$  as well. Clearly  $-d = -\sum_{t=1}^p d_t$  is orthogonal to this surrogate plane.

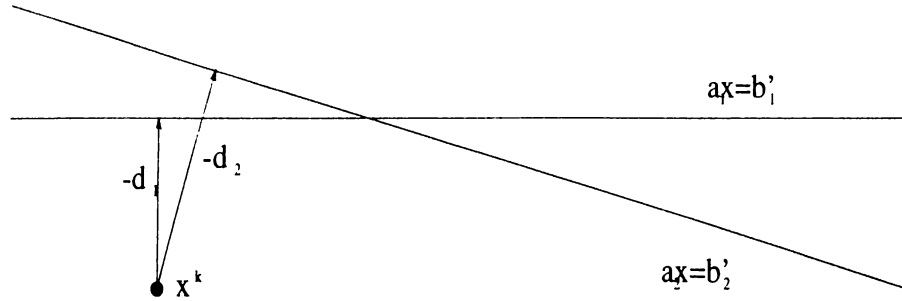


Figure 3.3: Two surrogate planes  $a_1 x = b'_1, a_2 x = b'_2$  and the movement directions induced by them.

This vector has the same direction with the one used in (3.17) when equal weights are assigned. Define  $-d_{F^k}$  (also orthogonal to the surrogate plane) so that:

$$P_{F^k}(x^k) = x^k - d_{F^k} \quad (3.22)$$

where  $P_{F^k}$  is the unrelaxed projection onto the surrogate plane. Let  $C^k$  represent the containing polyhedron  $\bigcap_{t=1}^p \{x : a_t x \leq b'_t\}$  and  $F^k$  denote the half space defined by the surrogate inequality  $\{x : \sum_{t=1}^p a_t x \leq \sum_{t=1}^p b'_t\}$ .

**Lemma 3.5.1**

$$-d_{F^k} = -\frac{\sum_{t=1}^p \|d_t\|^2}{\|\sum_{t=1}^p d_t\|^2} \sum_{t=1}^p d_t \quad (3.23)$$

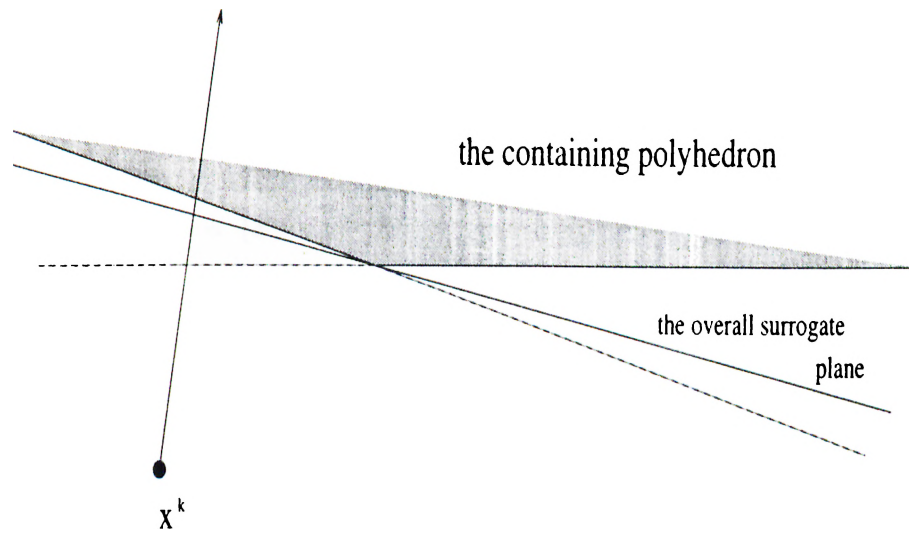


Figure 3.4: Movement in the direction of  $-(d_1 + d_2)$  orthogonal to the overall surrogate plane, which supports the containing polyhedron.

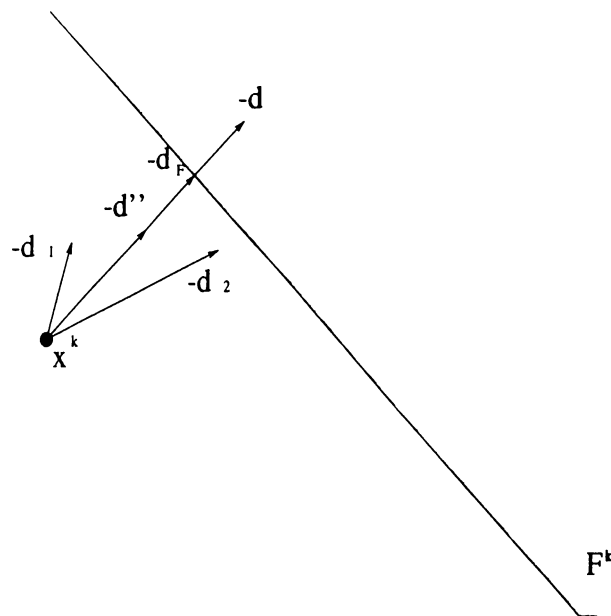


Figure 3.5: Unrelaxed projection onto overall surrogate plane. ' $x^k - d_{F^k}$ ' gives the unrelaxed projection onto this plane. ' $-d = -\sum d_i$ ' takes  $x^{k+1}$  beyond this plane. On the other hand with ' $-d'' = -\sum \tau_i d_i$ ' (where  $\sum \tau_i = 1$ )  $x^{k+1}$  does not quite reach the plane.

**Proof:** By definition  $\forall t$ ,  $P_t(x) = x - \left(\frac{a_t x - b'_t}{\|a_t\|^2}\right) a_t$  and  $d_t = \left(\frac{a_t x - b'_t}{\|a_t\|^2}\right) a_t$ . Rewrite  $d_t x = b_t$  instead of  $a_t x = b'_t$  by replacing  $a_t, b'_t$  with  $d_t, b_t$  respectively. Doing this we have:

$$d_t = \left(\frac{d_t x - b_t}{\|d_t\|^2}\right) d_t$$

which implies that  $d_t x - b_t = \|d_t\|^2$ . Indeed, ' $d_t x - b_t$ ' is the squared distance of  $x^k$  to the set  $C_t^k = \{x : a_t x \leq b'_t\}$ . Now, consider the overall surrogate plane  $\sum_{t=1}^p d_t x = \sum_{t=1}^p b_t$  and the unrelaxed projection onto it:

$$\begin{aligned} P_F(x^k) &= x^k - \left(\frac{\sum_{t=1}^p d_t x - \sum_{t=1}^p b_t}{\|\sum_{t=1}^p d_t\|^2}\right) \sum_{t=1}^p d_t \\ &= x^k - \left(\frac{\sum_{t=1}^p (d_t x - b_t)}{\|\sum_{t=1}^p d_t\|^2}\right) \sum_{t=1}^p d_t \\ &= x^k - \left(\frac{\sum_{t=1}^p \|d_t\|^2}{\|\sum_{t=1}^p d_t\|^2}\right) \sum_{t=1}^p d_t \end{aligned}$$

■

From **Lemma 3.5.1** we get the idea of the new step size. Disregarding relaxation, the conventional step given by equation (3.17) as the convex combination of distinct block steps in the simultaneous algorithm is very short and does not quite reach the overall surrogate plane given by equation (3.21) in practice. It is verified in [Kiwiel 95] that the long-step algorithms generate deeper surrogate cuts than the conventional short-step algorithms. Hence, the longer step in (3.20) has the most appropriate size, since it projects the current point exactly onto the overall surrogate plane. It is the longest step which yields a Fejér-monotonic sequence when utilized with a relaxation parameter between 0 and 2.

**Lemma 3.5.2** *Redefine  $P_F(x^k) = x^k - \lambda_k d_F$  where  $0 < \lambda_k < 2$  if  $x^k \notin K$  and let  $x^{k+1} = P_F(x^k)$ . Then the sequence  $\{x^k\}_{k=0}^\infty$  is strictly Fejér-monotone with respect to  $F^k$  (hence with respect to  $C^k$ , and  $K$ , since  $K \subseteq C^k \subseteq F^k$ ).*

**Proof:** Although obvious geometrically we provide an analytic verification.  $x^{k+1}$  is contained within the line segment between  $x^k$  and  $x^k - 2d_{F^k}$  (excluding the end points). Let  $x_{F^k}$  be the unrelaxed projection ( $\lambda_k = 1$ ) onto  $F^k$ , i.e.  $x_{F^k} = x^k - d_{F^k}$ . Let  $f$  be any point in  $F^k$ . As also seen from Figure 3.6:

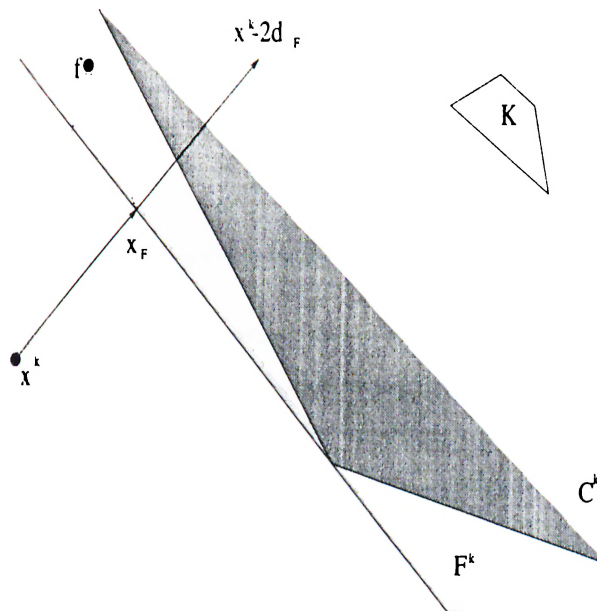


Figure 3.6: Illustration of the Fejér-monotonicity of the generated sequence. (Superscripts of some of the  $F$ 's are dropped for clarity.)

$$\|x^k - f\|^2 = \|(x^k - x_{F^k}) - (f - x_{F^k})\|^2$$

$$\|x^k - f\|^2 = \|x^k - x_{F^k}\|^2 + \|f - x_{F^k}\|^2 - 2(x^k - x_{F^k}) \cdot (f - x_{F^k}) \quad (3.24)$$

where the dot product term is nonpositive due to convexity of  $F^k$ . On the other hand,

$$\|x^{k+1} - f\|^2 = \|(x^{k+1} - x_{F^k}) - (f - x_{F^k})\|^2$$

$$\|x^{k+1} - f\|^2 = \|x^{k+1} - x_{F^k}\|^2 + \|f - x_{F^k}\|^2 - 2(x^{k+1} - x_{F^k}) \cdot (f - x_{F^k}) \quad (3.25)$$

If  $1 < \lambda_k < 2$  the dot product term in (3.25) is nonnegative (see Figure 3.6). Conversely, if  $0 < \lambda_k < 1$  the dot product term is nonpositive but since  $x^{k+1}$  is contained in the line segment between  $x^k$  and  $x_{F^k}$  we have  $(x^{k+1} - x_{F^k}) = \delta(x^k - x_{F^k})$  for some  $0 < \delta < 1$ . Hence in both cases,  $(x^{k+1} - x_{F^k}) \cdot (f - x_{F^k}) \geq (x^k - x_{F^k}) \cdot (f - x_{F^k})$ . Since  $\lambda_k < 2$ , we always have  $\|x^{k+1} - x_{F^k}\| > \|x^k - x_{F^k}\|$  which yields  $\|x^k - f\|^2 > \|x^{k+1} - f\|^2$  and hence,  $\|x^k - f\| > \|x^{k+1} - f\|$ . ■

**Remarks.** (1) It has been assumed that for the simultaneous block projections approach each infeasible block has the same weight, i.e.  $\tau_t = \frac{1}{p^k}$ ,  $\forall t, A^t x^k \not\leq b^t$  ( $p^k$  being the total number of infeasible blocks at iteration  $k$ ). This is quite plausible since the matrix  $A$  is sparse and unstructured. However, **Lemmas 3.5.1**, and **3.5.2**

are still valid when arbitrary (all positive) weights are assigned to infeasible blocks. In that case the projection onto the ‘weighted surrogate plane’ becomes:

$$-d_{F^k} = - \left( \frac{\sum_{t=1}^p \tau_t \|d_t\|^2}{\|\sum_{t=1}^p \tau_t d_t\|^2} \right) \sum_{t=1}^p \tau_t d_t \quad (3.26)$$

which can be similarly derived by replacing  $d_t x^k - b_t$  by  $\|d_t\|^2$  for all  $t = 1, \dots, p$ .

(2) Fejér-monotonicity of the generated sequence is still valid when the containing domain is an arbitrary convex superset  $S$  instead of a surrogate inequality, provided that a valid projection somehow exists and  $\lambda_k$  is between 0 and 2. In the cases of convex feasibility, movement directions might be chosen as combined subgradients as done in [Oettli 72], [Censor & Lent 82], [Dos Santos 87] and projections can be made onto valid separating hyperplanes.

Hence we can restate the ‘long-step block projections algorithm’ as follows:

### The Long-Step Block Projections Algorithm:

**Step 0.** Let  $A \in \mathfrak{R}^{m \times n}$  and  $b \in \mathfrak{R}^m$ . Consider the even partitioning of the  $A$  matrix and the  $b$  vector into  $p$  rowwise blocks of almost equal sizes, as  $[A^1|A^2|\dots|A^p]^T$  and  $[b^1|b^2|\dots|b^p]^T$ . (In the case of parallel implementation  $p$  is equal to the number of processors.) Let  $x^0 = 0$  and  $k = 0$ .

**Step 1.** For  $t = 1, \dots, p$ , based on a selection of  $\tau_t$  values (satisfying  $\forall \tau_t \geq 0$  and  $\sum_{t=1}^p \tau_t = 1$ ) check if  $A^t x^k \leq b^t$  for all  $t$  with  $\tau_t > 0$ . If so, then let  $d_t^k = 0$ . Otherwise let

$$d_t^k = \frac{(\pi_i^k A^t x^k - \pi_i^k b^t)(\pi_i^k A^t)}{\|\pi_i^k A^t\|^2}$$

where  $\pi_i^{t,k} > 0$  if constraint  $i$  is violated and  $\pi_i^{t,k} = 0$  otherwise. The next point is generated as:

$$x^{k+1} = x^k - \lambda_k \left( \frac{\sum_{t=1}^p \tau_t \|d_t^k\|^2}{\|\sum_{t=1}^p \tau_t d_t^k\|^2} \right) \sum_{t=1}^p \tau_t d_t^k \quad (3.27)$$

Update the total number of violated constraints, in all blocks.

**Step 2.** If the total number of violated constraints in the iteration is zero then stop, the current solution is feasible. Otherwise, assign zero to the number of violated constraints, let  $k = k + 1$  and go to **Step 1**.

In this modified algorithm one will have longer steps when it is implemented with simultaneous projections. Note that if the algorithm is implemented with successive projections (cyclic control) the iterative update (3.27) will reduce to  $x^{k+1} = x^k - \lambda_k d_t^k$ . One should recall that, the successive versions of the conventional and long-step block projections algorithms are identical, hence the improvement is valid only for the simultaneous block projections case, when the algorithms differ.

**Lemmas 3.5.1** and **3.5.2** establish that the modified simultaneous algorithm can be viewed as a basic surrogate constraint algorithm which generates a separating hyperplane at each major iteration and the direction vector given in equation (3.23) (or (3.26)) provides the unrelaxed projection onto this valid hyperplane through  $P_{F^k} = x^k - d_{F^k}$  (see Figure 3.6). Hence, with this interpretation, this algorithm belongs to the general class of ‘projection-onto-separating-hyperplane algorithms.’

In retrospect, it can be seen that this algorithm falls into the framework of the generalized algorithm with long steps, outlined by equations (3.10) through (3.16) in [Kiwiel 95]. Rewriting equation (3.27) as  $x^{k+1} = x^k - \lambda_k(\sigma_k/\tilde{\sigma}_k)\sum_{t=1}^p \tau_t d_t$  (so that  $(\sigma_k/\tilde{\sigma}_k)$  is equivalent to  $\left(\frac{\sum_{t=1}^p \tau_t \|d_t\|^2}{\|\sum_{t=1}^p \tau_t d_t\|^2}\right)$ ), it is verified in *Lemma 4.3 (iii)* of [Kiwiel 95] that  $(\sigma_k/\tilde{\sigma}_k) \geq 1$  which indicates that the long-step method uses the same movement direction as the one used in the short-step method with larger step sizes.

**Theorem 3.5.1** *The block projections algorithm with the modified step as given by equation (3.27) converges to a point in  $K$ , provided that it is nonempty.*

The verification of the theorem follows from the fact that the algorithm falls in the category of long-step methods outlined by (3.10) and (3.15) in [Kiwiel 95]. By *Theorem 3.11* of [Kiwiel 95], the algorithm generates sequences which converge to  $K$ .

### 3.6 Implementation Results for Random Problems

In this section, test results in terms of iterations are given for five routines: (i) the successive Yang-Murty block projections algorithm, (ii) the simultaneous

$p //$ $m. n. \text{ int.}$	2 //	4 //	8 //	16 //
500. 1000, 0.02	15.4(7.2)	27.8(6.2)	51.8(5.6)	104.6(5.6)
2000. 1000, 0.02	119(59)	203(50)	365.4(44.8)	651.8(39.8)
5000. 2500, 0.02	106.6(52.8)	191.8(47.2)	367(45)	712.6(43.6)
10000. 5000, 0.01	110.2(54.6)	205.4(50.6)	392.6(48.2)	754.2(46.2)
20000. 10000, 0.002	144.6(71.8)	261.4(64.6)	479(59)	885.4(54.4)
50000. 20000, 0.001	290.2(144.6)	508.6(126.4)	941.4(116.8)	1775(110)

Table 3.1: Average number of block and major iterations (numbers in parentheses represent the major iterations) of the sequential algorithm of Yang and Murty. 'int.' stands for the nonzero intensity of the matrix.  $m$  and  $n$  are the row and column sizes respectively and  $p$  represents the number of blocks. Five test problems have been solved for each size.

Yang-Murty block projections algorithm, (iii) the simultaneous Yang-Murty block projections algorithm with a step size magnified by a factor equal to the current number of violated blocks (iv) the simultaneous Yang-Murty block projections algorithm with the Gar.-Pal.-Gon.-Cas. step. (v) the simultaneous Yang-Murty block projections algorithm with the modified step suggested in equation (3.20).

The figures in the tables represent major (full) iterations for the simultaneous algorithms and block and major iterations for the successive algorithm. To compare the performance of the simultaneous algorithm with that of the successive algorithm based on the iteration results, one should assume that (in the ideal case) a block iteration of the successive algorithm is more or less equivalent to a full iteration of the simultaneous algorithm; which is not indeed the case, as can be seen from the timing results. However, while comparing different instances (by varying  $p$ ) of the successive algorithm, the figures representing the major iteration should be considered.

We have tabulated timing results for the successive block projections algorithm of Yang-Murty (Table 3.6) and the simultaneous block projections algorithm with the modification described in Section 3.5 (Table 3.7). Only the largest 3 problem sets are tabulated since for smaller problems convergence occurs in a few seconds. We have not tabulated results for the original simultaneous block projections algorithm of Yang-Murty and other parallel routines since the iterative results in Table 3.2 are already much worse than those of the successive block projections algorithm. The



$p //$ $m, n, \text{int.}$	2 //	4 //	8 //	16 //
500, 1000, 0.02	27.6	69.4	209.4	507
2000, 1000, 0.02	267.4	1422.8	3393.8	6921.4
5000, 2500, 0.02	1087.6	3274.8	6524.2	13069.2

Table 3.2: Average number of major iterations of the original parallel algorithm suggested by Yang and Murty.  $p$  represents the number of blocks and hence the processors. Same test problems used in the sequential experiments are used, but the largest 3 problem sets are not solved since the number of iterations become prohibitive. Comparison of these results with those of the previous table indicates that the simultaneous version of the block projections algorithm with short steps, is quite poor in performance.

$p //$ $m, n, \text{int.}$	2 //	4 //	8 //	16 //
500, 1000, 0.02	7.2	7	6.6	$\bar{7}$
2000, 1000, 0.02	84	152.2	-	-
5000, 2500, 0.02	86.6	979.6	-	1164.6

Table 3.3: Average number of major iterations of an implementation of the parallel Yang-Murty algorithm with the magnified step size without premultiplication with the regulative parameter. Same test problems as in the previous table are used. '-' represents a typical case where the routine does not converge to a feasible point. It is seen that there is some improvement (when compared to the parallel version of the Yang-Murty algorithm) for problems with relatively small sizes but performance is still poor for larger problems. Furthermore, the algorithm does not always converge.

$p //$ $m, n, \text{int.}$	2 //	4 //	8 //	16 //
500, 1000, 0.02	24.2	24.2	26.6	27.8
2000, 1000, 0.02	193	200.2	193.4	197.8
5000, 2500, 0.02	612.8	643	646.6	669

Table 3.4: Average number of major iterations for an implementation of the parallel algorithm with the step suggested by Gar.-Pal.-Gon.-Cas. Same test problems as in the previous experiments given in the previous two tables have been used. The results are better when compared to the parallel Yang-Murty algorithm, but are still quite bad when compared to the sequential Yang-Murty algorithm.

$p //$ $m, n, \text{int.}$	2 //	4	8 //	16 //
500, 1000, 0.02	7.4	6.8	7.2	6.6
2000, 1000, 0.02	66.8	62.8	59.4	53.8
5000, 2500, 0.02	66	65.6	65	63
10000, 5000, 0.01	69.8	69	68	66.6
20000, 10000, 0.002	80.6	77.8	74.6	69.2
50000, 20000, 0.001	180.2	172.6	166.2	158.4

Table 3.5: Average number of major iterations for an implementation of the long-step simultaneous block projections algorithm. Same test problems as in the experiments given in the previous tables are used. The simultaneous algorithm with the improved step size performs better than the sequential Yang-Murty algorithm, under the assumption that a full iteration of the simultaneous routine takes about the same time as a block iteration of the successive routine.

$p //$ $m, n, \text{int.}$	2 //	4 //	8 //	16 //
10000, 5000, 0.01	29.6	28.6	27.8	29
20000, 10000, 0.002	36	34.2	34.2	36
50000, 20000, 0.001	189.2	170.8	171.2	183.4

Table 3.6: Average processing times (in wallclock seconds) for the successive Yang-Murty algorithm. Same test problems (the largest 3 problem sets) are used.

distributed implementation of the simultaneous algorithm has been realized by the aid of PowerPVM on a Parsytec CC-24 machine. Timing results have been obtained as wallclock seconds. CPU results are not representative since message transfers between the processors are neglected by CPU timers. The timing results, although not as good as the iteration results (due to the communication overhead between the parallel processors), are still very encouraging and indicate that the parallel implementation of the long-step simultaneous block projections algorithm is quite competitive over the successive block projections algorithm.

$p //$ $m, n, \text{int.}$	2 //	4 //	8 //	16 //
10000, 5000, 0.01	19.6	12	9.2	12
20000, 10000, 0.002	21.2	13.4	11.2	13
50000, 20000, 0.001	121.6	69.2	47	43.4

Table 3.7: Average processing times (in wallclock seconds) for the long-step simultaneous block projections algorithm. Same test problems with those given in the previous table are used. The timing results are better than those obtained by the sequential Yang-Murty algorithm, although we do not have as much improvement as indicated by a comparison of the iteration results.

$p //$ $m, n, \text{int.}$	2 //	4 //	8 //	16 //
10000, 5000, 0.01	1.51, 0.76	2.38, 0.60	3.02, 0.38	2.42, 0.15
20000, 10000, 0.002	1.70, 0.85	2.55, 0.64	3.05, 0.38	2.77, 0.17
50000, 20000, 0.001	1.56, 0.78	2.47, 0.62	3.64, 0.46	4.23, 0.26

Table 3.8: Speedup and efficiency measures of the new parallel implementation. The figures in the cells represent speedups and efficiencies respectively.

### 3.7 Notes on Implementation

In this section we discuss issues related to random generation of the test problems and parallel implementation on distributed computers.

#### 3.7.1 Implementation Data and Test Parameters

It is assumed that the matrix underlying the feasibility problem is sparse. This matrix is stored in rowwise format to ease its access. Using this type of data storage, the matrix-vector products can be computed easily. The widely known standard sparse matrix formats are used [Pissa. 84], but to ease the control over the storage arrays, we have added one more cell to each, which marks the end of the array.

The sample test problems have been generated as follows: First of all, a matrix with a given size and sparsity percentage is generated so that the nonzero elements are distributed uniformly and such that each nonzero value is uniformly distributed

between  $-5.0$  and  $5.0$ . The random distribution has been realized in two ways. In the first, an exact number of nonzero entries is fixed and they are distributed uniformly (with parameters depending on the problem size) to the rows of the matrix so that each row has at least one nonzero element. Then, their column numbers are generated uniformly. In the second, a simulated sequence of a Poisson process is obtained and points are generated with exponential interarrival times with a parameter value equal to the nonzero intensity of the matrix. The exponential density is truncated between 1 and  $n$  and the generated interarrival time is rounded to the nearest integer and the next nonzero entry position is found by adding this value to the column number of the previously placed nonzero entry. When one row is finished, the process is continued in the next row. This approach is somewhat better than the first, since the nonzero entries are generated sequentially and one does not require a reordering when storing the data in rowwise format. Here we utilize the property stating that in a Poisson process, given that  $n$  arrivals have occurred within a time interval  $(0, t)$ , the distribution of the arrival times  $S_1, \dots, S_n$  have the same distribution as the order statistics of  $n$  independent random variables distributed on this interval  $(0, t)$  (see for example [Ross 83]). We assume that this idea is applicable to a discrete interval (the entire matrix in row stacked vector format, being visualized as a very long discrete time interval) and that the truncation of the exponential distribution does not have any significant effect on the uniformity of the nonzero distribution throughout the matrix.

After generating the random matrix, a random vector  $x$  is generated, such that each of its elements lie in the interval  $(-4.5, 4.5)$ . Following this,  $Ax$  is computed and the vector  $b$  is generated according to  $b_i = A_i x + u_i$ , where  $u_i$  is a discrete 0-1 uniform random variable. In this way a feasible polyhedron is created. Our aim is to keep this polyhedron somewhat small and distant to the initial point of the algorithm, so that trivial convergence in a few steps will not occur.

An important issue is the selection of the weight vector  $\pi^{t,k}$ . Weights within a block may be distributed equally among all violated constraints or they can be assigned in proportion to the amount of violations. A suitable combination of the two approaches may also be used. In our tabulated results we have used the hybrid approach (which has also been used in [Yang & Murty 92]):

$$\pi_i^{t,k} = \frac{0.2(A_i^t x^k - b_i)}{\sum_{h:A_h^t x^k > b_h^t} (A_h^t x^k - b_h^t)} + \frac{0.8}{\text{number of violated constraints}} \quad (3.28)$$

For convenience it is assumed that  $\sum \pi_i^{t,k} = 1$ . The relaxation parameter  $\lambda_k$  has been fixed at 1.7 and the feasibility tolerance ( $\varepsilon$ ) is taken as  $10^{-9}$  (to enable finite convergence).

### 3.7.2 Parallel Architecture

In parallel implementation the number of submatrix blocks should be equal to the number of processors. Each processor deals with a single block, so that it is quite natural to divide the matrix into  $p$  submatrices as evenly as possible, so that each submatrix has about  $\lceil \frac{m}{p} \rceil$  rows. Each processor checks its constraints (which are equal or almost equal in size) for feasibility and computes the projection if necessary. When a processor finishes its task, it waits for the others to finish as well.

During the subroutine operations (feasibility checks and projection calculations for distinct blocks) each processor works independently and no message passing among the machines is required. Then, the new iteration point is calculated according to equations (3.9) or (3.20). After this step, the new point is broadcasted to all processors and the procedure is repeated until a feasible point is found.

The broadcasting of the vector  $x^k$  (at the beginning of the iteration) and the direction vectors  $d_1, d_2, \dots, d_p$  among all processors to compute  $x^{k+1}$  (at the end of the iteration) is a critical issue. A natural way to accomplish this is the following:  $x^k$  is transmitted to all processors from the first processor. Later, each machine computes its corresponding direction vector  $d_t$  and sends it to the first processor where  $x^{k+1}$  is computed (see Figure 7). When the number of processors are large (8 or 16), this procedure becomes quite inefficient due to the overload burden resulting from the transmission of these arrays of sizes ranging up to 20,000 double precision numbers.

To overcome this inefficiency, we have considered another design. Let each processor  $t$  compute the related  $d_t$  and also  $\|d_t\|^2$  for the given  $x^k$ . Computing  $\sum_{t=1}^p \|d_t\|^2$  is trivial, since it only involves a global sum operation (over the  $\|d_t\|^2$  values). For  $\sum_{t=1}^p d_t$ , let each machine be responsible for computing a fractional part ( $1/p$ ) of the summation vector. Thus it is sufficient to broadcast only the fractional part of the related direction vectors. The quantity  $\|\sum_{t=1}^p d_t\|$  is also determined by global summation over the fractional summed parts. Naturally, each

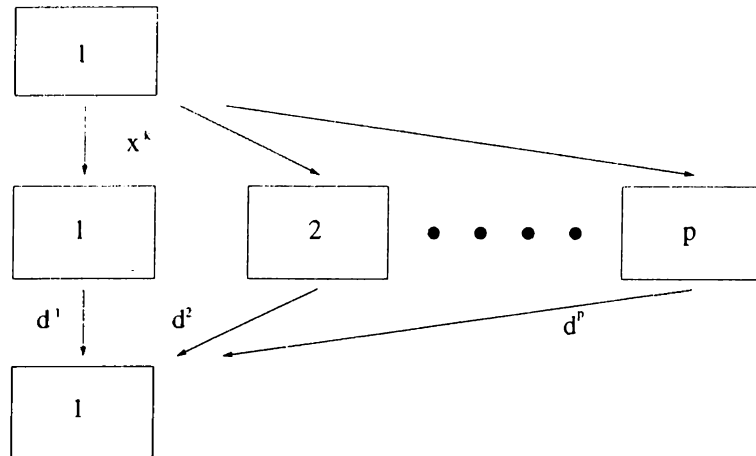


Figure 3.7: Conventional parallel architecture.

processor will update the corresponding portion of  $x^k$  with respect to equation (3.20). The fractional portions of  $x^k$  (which are  $1/p^{\text{th}}$  of the original vector size) should be broadcasted to all machines. The ‘fold’ (over  $d_t$ ) and ‘expand’ (over  $x^k$ ) operations just outlined, avoid the transmission of very large vectors among the processors, which would greatly slow down the parallel implementation. A more sophisticated parallel implementation of the long-step block projections algorithm has been recently reported in [Turna 98].

The initialization phase is carried out on each processor independently to avoid further communication among the blocks. If a test problem is to be randomly generated, the initial seed is broadcasted to all machines. If the initial seed has to be read from a data file, this is done by all machines separately.

The distributed implementation of the algorithm has been developed with the aid of PVM 3.3.11 on several Sparc workstations. The algorithm is governed by a main C routine, which makes calls to a C subroutine (for the parallel block operations) and to several PVM functions (see [Geist *et al.* 94]). For wallclock timings, the same routine with slight modifications has been compiled and executed by the aid of PowerPVM/EPX on a Parsytec CC-24 machine.

### 3.8 Generalization of the New Step Size to Convex Feasibility Problems and Subgradient Methods

The idea of the longer step regulated by the fractional parameter utilized in (3.20), can be easily generalized to the case of convex feasibility. Note that the Fejér-monotonicity proofs are valid for any valid separation of the test point  $x^k$  and  $K$ . This may be a separating hyperplane or even a convex superset containing  $K$ . Here we will restrict ourselves to the case of separating hyperplanes.

We first outline the cyclic subgradient projections method (CSP) in [Censor & Lent 82]. This is an algorithm of successive projections onto valid separation domains, as also illustrated in [Dos Santos 87]. The underlying feasibility problem is of the form (identical to the one that will be posed in Chapter 4):

$$K = \{x \in \mathbb{R}^n : f_i(x) \leq 0\} \quad (3.29)$$

where the  $f_i$  for all  $i$  are convex throughout the applicable domain. For convenience we assume differentiability as well. Let  $g_i$  represent the gradient of  $f_i(x)$ . The CSP algorithm is given as:

**Step 0.**  $x^0$  arbitrary,  $0 < \lambda_k < 2$ , for all  $k$ .

**Step 1.**  $x^{k+1} = x^k - \lambda_k \frac{f_i(x^k)}{\|g_i\|^2} g_i$

Now, we present the parallel version of this step. Here we consider simultaneous projections onto all separating planes (orthogonal to the gradient directions). Thus, the simultaneous step becomes (we consider the case of assigning equal weights to the individual steps):

$$x^{k+1} = x^k - \lambda_k \left( \frac{\sum_i f_i(x^k)}{\|\sum_i g_i\|^2} \right) \left( \sum_i g_i \right) \quad (3.30)$$

In general, the same idea is also applicable to the  $(\delta, \eta)$  approach mentioned earlier (and actually to any simultaneous approach ranging from Cimmino's algorithm to the most sophisticated block projections method). The key is to replace the short step determined as the convex combination of individual projections with the long step given in (3.26).

## Chapter 4

# The Convex Feasibility Problem

In this chapter we will first describe the central cutting plane algorithms based on the routines of [Ye 89], [Gof. *et al.* 93a, b]. Then we propose a variation of these methods which utilize rectangular cuts and subgradients as movement directions.

The cutting plane methods utilize the analytic center of the polytope as a test point. As stated previously, central test points are essential since central cuts induce faster shrinkage of the containing polytope even if the most unfavorable hyperplane is selected. So before going into these algorithms, a brief overview of various definitions for the center of convex bodies will be given in the next section. In Section 4.2 the Ye-Goffin approach is summarized. In Section 4.3 the effect of multiple cuts is discussed. In Section 4.4 average convergence behavior of a typical central cutting plane approach is analyzed.

In the later Sections 4.5, 4.6, and 4.7 a new algorithm which maintains a containing box and (at the same time) utilizes subgradient directions is developed. In Section 4.8 some geometrical and analytical concepts for analyzing this algorithm are given. In Section 4.9 convergence of the algorithm is established.



## 4.1 A Variety of Centers

The most natural center is the center of gravity of the convex domain. Indeed it is known that the center of gravity cuts lead to high shrinkage rates. It has been shown by Grünbaum and Mityagin that the volumes of the successive polytopes satisfy the following inequality (in  $\mathfrak{R}^m$ ):

$$V(\Omega^{k+1}) \leq \left[1 - \left(1 - \frac{1}{m+1}\right)^m\right] V(\Omega^k) \quad (4.1)$$

Determining the center of gravity of a polytope is complicated, so despite this nice bound for shrinkage rate, a simpler center is necessary. In [Tarasov *et al.* 88] the polytope is approximated by an inscribed ellipsoid of maximal volume. If valid cuts through the centers of these ellipsoids are introduced, then convergence of the algorithm is quite good. However, determining this center for arbitrary polytopes is still complicated, although possible in polynomial time [Khachiyan & Todd 90], [Tarasov *et al.* 88].

The analytic center proposed by Sonnevend is much easier to compute and several polynomial time routines are given through Newtonian subroutines (see for example [Renegar 88]). The analytic center of a polytope  $\Omega = \{y : A^T y \leq c\}$  is defined as the unique maximizer of the function:

$$\phi(y) = \sum_{j=1}^n \ln(c - A^T y)_j \quad (4.2)$$

Another center which has gained some interest, is the volumetric center of [Vaidya 89]. Computation of the volumetric center is more efficient than the center of the maximal inscribed ellipsoid defined above [Atkinson & Vaidya 95]. It is calculated as the center of the maximal inscribed ellipsoid as well, but in this case a restricted class of ellipsoids are taken into account. According to this restriction, only ellipsoids which have axes parallel to the coordinate axes are considered. It is defined as the minimizer of the following potential function:

$$F(y) = \frac{1}{2} \ln(\det(H(y))) \quad (4.3)$$

where  $H(y)$  is the Hessian of the negated potential function of the analytic center:

$$- \phi(y) = - \sum_{j=1}^n \ln(c - A^T y)_j \quad (4.4)$$

and is positive definite  $\forall x \in \text{int}(\Omega)$ . The maximal inscribed ellipsoid of the polytope  $\Omega$  is given as:

$$E(H^{-1}(y^0), y^0, 1) = \{y : (y - y^0)^T H(y^0)(y - y^0) \leq 1\} \quad (4.5)$$

where  $y^0$  is the volumetric center computed as above. It has been shown that when cuts are introduced from the volumetric center, the volumes of successive ellipsoids satisfy:

$$V(E^{k+1}) \leq 0.843V(E^k) \quad (4.6)$$

A detailed discussion of centers may be found in [Kaiser *et al.* 91].

## 4.2 The Ye-Goffin Approach of Analytic Centers

In this section we give a short discussion of the possible improvements that can be made on the convex feasibility model described in [Ye 89] and [Gof. *et al.* 93a, b]. The problem can be implicitly defined by a separation oracle at a given test point. Each iteration of the Ye-Goffin algorithm gives a separating hyperplane between the test point and the convex set. The procedure is repeated until an interior point of the convex body is found. The selection of the test point is essential for rapid volume shrinkage of the containing polytope and hence the rapidity of the algorithm. As previously stated, the center of gravity cuts lead to algorithms which have relatively high shrinkage rates; however computation of the center of gravity is too complicated. Thus, in the recent literature the use of analytic centers has become popular. The cuts obtained from the analytic centers are satisfactory [Gritz. & Klee 93a] and can be computed approximately with the desired tolerance within a reasonable amount of Newton steps.

The convex feasibility problem posed in [Gof. *et al.* 93a, b] can be described briefly as follows:  $\Gamma$  is a convex set defined implicitly by a separation oracle. Given any point  $\bar{y}$ , the oracle either answers that  $\bar{y} \in \text{int}(\Gamma)$  or generates a separating hyperplane  $a^T(y - \bar{y}) = 0$ ,  $|a| = 1$  and  $a^T y \leq a^T \bar{y}$ , which is valid for  $\Gamma$ . It is assumed that  $\Gamma$  is contained in the unit cube, and it contains in itself a full dimensional ball of radius  $\epsilon$  (hence its interior is nonempty). In our analysis,  $\Gamma$  is explicitly expressed as:

$$\Gamma = \{y \in \mathfrak{R}^m : f_i(y) \leq 0, i = 1, 2, \dots, p\}$$

such that  $f_i$ 's are convex. The problem is solved when a suitable  $y \in \Re^m$  such that all  $f_i(y) < 0$  is determined.

Central cutting algorithms start with an elementary containing domain, a ball, a cube or a simplex for which the center is trivial or easily computable. A valid cut passing from this point is generated and the new center of the shrunk domain (polytope, ellipsoid or whatever) is approximated or computed with a high degree of accuracy (or directly calculated if it is trivial). The procedure is repeated and the algorithm stops when an interior point is found. Some classes of these algorithms also allow for infeasible inputs and terminate appropriately if such cases are encountered.

Now we give the algorithm in [Gof. *et al.* 93a, b]:

### The Cutting Plane Algorithm of Analytic Centers:

#### Step 0.

$$\begin{aligned} A^0 &= (I, -I) \in \Re^{m \times 2m} \\ c^0 &= \begin{pmatrix} e \\ 0 \end{pmatrix} \in \Re^{2m} \\ \Omega^0 &= \{y \in \Re^m : 0 \leq y \leq 1\} \\ y^0 &= \frac{1}{2}e \in \Re^m, \quad s^0 = \frac{1}{2}e \in \Re^{2m}, \quad x^0 = 2e \in \Re^{2m} \end{aligned}$$

**Step 1.** Find the center  $y^k$  of  $A^k$ . Check if  $y^k \in \text{int}(\Gamma)$  and stop in that case. If not, choose any  $i$  such that  $f_i(y^k) \geq 0$ . Let  $g_i$  be a subgradient of  $f_i$  at  $y^k$ , and let  $a_{k+1} = \frac{g_i}{\|g_i\|}$ . Then  $\{y : a_{k+1}^T y \leq a_{k+1}^T y^k\} \supset \Gamma$ , so let  $A^{k+1} = (A^k, a_{k+1})$ ,  $c^{k+1} = \begin{pmatrix} c^k \\ a_{k+1}^T y^k \end{pmatrix}$  and  $\Omega^{k+1} = \{y \in \Re^m : (A^{k+1})^T y \leq c^{k+1}\}$ .

**Step 2.**  $k = k + 1$ , go to **Step 1**.

The implicit oracle definition provides a column generation scheme for the Ye-Goffin approach. Since we assume that  $\Gamma$  is defined as in the previous section, calling the oracle amounts to a sequential check of those inequalities. In the following section, we will consider introducing all (or some) of the cuts generated from the violated inequalities one at a time. But for particular problems, each call of the oracle might be quite costly depending on the prespecified implicit definition, so that this approach might become unimplementable.

Recalling the analytic center of the polytope  $\Omega = \{y : A^T y \leq c\}$  as the point  $\bar{y}$  maximizing  $\phi(y)$  in (4.2), the potential function of the polytope will be defined as:

$$P(\Omega) = \phi(\bar{y}) = \max_{y \in \Omega} (\phi(y)) \quad (4.7)$$

This algorithm guarantees positive reduction of the potential value at each iteration with appropriate parameters [Ye 89], [Gof. *et al.* 93a].

The interest in  $\Omega$ , the containing polytope in the *dual space*, has arisen from the availability of the centering direction which has been defined originally for primal-dual path following methods. Let us consider the following primal-dual pair of linear programs for an instant:

$$\begin{aligned} (P) \quad & \min \quad c^T x & (4.8) \\ & \text{s.t.} \quad Ax = b \\ & \quad \quad x \geq 0 \end{aligned}$$

$$\begin{aligned} (D) \quad & \max \quad b^T y & (4.9) \\ & \text{s.t.} \quad A^T y + s = c \\ & \quad \quad s \geq 0 \end{aligned}$$

An interior point algorithm which stays close to the central path, has iteratively a better convergence rate than that of the projective scaling algorithm. This *central path* is the set of minimizers of the following problem with the logarithmic barrier function and the penalty term  $\mu > 0$  [Mont. & Adler 89]:

$$\begin{aligned} (P_\mu) \quad & \min \quad c^T x - \mu \sum_{j=1}^n \ln x_j & (4.10) \\ & \text{s.t.} \quad Ax = b \\ & \quad \quad x > 0 \end{aligned}$$

which has a unique solution  $x(\mu)$ . The necessary and sufficient conditions for  $(P_\mu)$  [Todd & Ye 90], [den Hertog *et al.* 90] are:

$$Xs - \mu e = 0 \quad (4.11)$$

$$Ax = b \quad (4.12)$$

$$A^T y + s = c \quad (4.13)$$

where  $X$  is the diagonal matrix version of the vector  $x$ . For a previous choice of  $s = \mu X^{-1}e$  (or  $\mu = \frac{x^T s}{n}$ ), one can rewrite (4.11) as  $Xs - \frac{x^T s}{n}e = 0$ . In fact, the set  $\mathcal{C} = \{(x, s), x \in \mathfrak{R}^n, s \in \mathfrak{R}^n : Xs = \frac{x^T s}{n}e\}$  is known as the central path. Path following methods move along the vicinity of this set to solve the original linear program. The duality gap for feasible  $x, y, s$  between (P) and (D) is  $c^T x - b^T y = x^T s = n\mu$ .

The conditions (4.11), (4.12), and (4.13) are all applicable for the following barrier function problem:

$$(D_\mu) \quad \max \quad b^T y + \mu \sum_{j=1}^n \ln s_j \quad (4.14)$$

$$\text{s.t.} \quad A^T y + s = c$$

$$s > 0$$

with the extra condition that  $x > 0$  [Todd & Ye 90].

The centering direction for the Newtonian subroutine in the main algorithm in [Gof. *et al.* 93b] is given as:

$$Sdx + Xds = \frac{x^T s}{n}e - Xs \quad (4.15)$$

$$Adx = 0 \quad (4.16)$$

$$A^T dy + ds = 0 \quad (4.17)$$

with  $d = (dx, dy, ds)$ . This direction is the Newton step from  $(x, s)$  to the central path with duality gap  $x^T s$ , for the original (P) and (D) LP pairs [Mizuno *et al.* 90].

This Newton subroutine is convergent whenever the starting point satisfies  $\|Xs - e\| \leq \gamma < 1$ , and  $x > 0, s > 0$  [Ye 89], [Gof. *et al.* 93b]. This is similar to the convergence criterion for the Newtonian minimization of the algorithm given in [Renegar 88].

When a new column  $a$  is added to  $A^k$  (a new cut included in  $\Omega^k$ ), such an  $(x, y, s)$  satisfying the above necessity condition can be found from the previous approximate center (of the previous polytope);  $(x^k, y^k, s^k)$  [Gof. *et al.* 93b]:

$$r^k = \sqrt{a^T (A(X^k)^2 A^T)^{-1} a} \quad (4.18)$$

$$\Delta y = -(\beta/r^k)(A(X^k)^2 A^T)^{-1} a \quad (4.19)$$

$$\Delta s = (\beta/r^k)A^T(A(X^k)^2A^T)^{-1}a \quad (4.20)$$

$$\Delta x = -(\beta/r^k)(X^k)^2A^T(A(X^k)^2A^T)^{-1}a \quad (4.21)$$

and,

$$y = y^k + \Delta y \quad (4.22)$$

$$s = \begin{bmatrix} (s^k + \Delta s) \\ (\beta r^k) \end{bmatrix} \quad (4.23)$$

$$x = \begin{bmatrix} (x^k + \Delta x) \\ (\beta/r^k) \end{bmatrix} \quad (4.24)$$

Starting from  $(x, y, s)$ , after a sufficient number of steps as given by (4.15), (4.16), and (4.17), the center can be found within a certain tolerance.

### 4.3 Multiple Cuts

By introducing many cuts at a test point (generated for all or part of the violated inequalities) or furthermore, calling the oracle at many test points inside the containing polytope (whenever implementing the oracle is not too costly), one could possibly obtain a better shrinkage performance, as also pointed out in [Ye 89].

Confining ourselves to a single test point at each iteration, we can include all cuts generated by all inequalities that hold as  $f_i(y^k) \geq 0$ , where  $y^k$  is the current test point. Note that each such inequality generates a valid cut of the form  $g_i y \leq g_i y^k$ . One might expect this approach to lead to a faster algorithm, especially for cases where the number of inequalities defining  $\Gamma$  is numerous. However, center updating at each iteration would be tedious and maintaining the containing polytope would become more difficult.

So we consider another way of determining the next test point. One can take a large single step into the next polytope. A possible movement direction is the negative sum of the normalized gradients of  $f_i(y^k)$ 's such that  $f_i(y^k) \geq 0$ , i.e. all of the violated inequalities which are taken into consideration. This direction will be denoted shortly as the NSNG direction. In mathematical terms it is expressed as  $\frac{\sum g_i}{\|\sum g_i\|}$ , under the assumption that all  $g_i$ 's are normalized. An important problem is to decide on the step length. At a given iteration the next test point is found as:

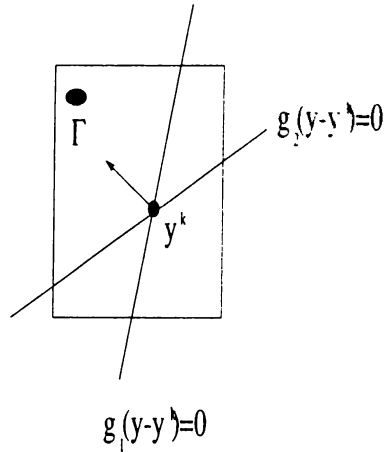


Figure 4.1: The NSNG direction for the case of two valid inequalities. The normal vectors  $g_1$  and  $g_2$  (each assumed to be of unit length) which define the cutting hyperplanes, are also illustrated. The NSNG direction is  $-(g_1 + g_2)$ .

$$y^{k+1} = y^k - \alpha \frac{\sum g_i}{\|\sum g_i\|} \quad (4.25)$$

where  $\alpha$  is the step size.

For appropriate recentering, one would naturally try to estimate the radius of the current polytope. As also discussed in [Gritz. & Klee 93b], this problem is very complicated even for simple polytopes. One could have considered the half-width of the new polytope if this was not the case. In this way the new test point would have been somewhat closer to the center. One could consider the width of the maximal inscribed ellipsoid, instead. Although this can be computed in polynomial time [Khachiyan & Todd 90], it is too complicated for practical purposes. One approach is to be satisfied with the volumetric center of the current polytope, which is easier to compute. (The notion of volumetric center has been introduced in the previous section.) This center can also be determined with a certain precision in polynomial time, but still not very efficiently [Gonzaga 92]. Thus the novel approach described here, is conceptual rather than practical.

The determination of the volumetric center and the width of the related ellipsoid will be illustrated. We can recall from equation (4.3) that,  $F(y) = \frac{1}{2} \ln(\det(H(y)))$ , where  $H(y)$  is the Hessian of  $\phi(y)$ , the negative logarithmic sum of the slack components. The volumetric center as stated, is the minimizer of  $F(y)$  over the polytope [Vaidya 89]. One can approximately obtain it through the Newtonian

routine with the following iteration:

$$y^+ = y - \gamma Q^{-1}(y) \nabla F(y) \quad (4.26)$$

where,

$$Q(y) = \sum_{j=1}^n \sigma_j(y) \frac{a_j a_j^T}{(a_j^T y - c_j)^2} \quad (4.27)$$

$$\nabla F(y) = \sum_{j=1}^n \sigma_j(y) \frac{a_j}{(c_j - a_j^T y)} \quad (4.28)$$

and,

$$\sigma_j(y) = \frac{a_j^T H^{-1}(y) a_j}{(a_j^T y - c_j)^2}, \quad 1 \leq j \leq n \quad (4.29)$$

in which  $a_j^T$  is the  $j^{\text{th}}$  row of  $A^T$  of the current polytope. As a starting point of this subroutine, we need an interior point of the new polytope. So we can start with, for example:

$$y = y^k - (2\epsilon) \frac{\sum g_i}{\|\sum g_i\|} \quad (4.30)$$

where  $y^k$  is the previous test point and  $\epsilon$  is the radius of the ball contained (by definition) in  $\Gamma$ .

Having computed the volumetric center  $y^0$  with a certain tolerance, the width of the related ellipsoid  $E(H^{-1}(y^0), y^0, 1)$  is  $2\sqrt{\lambda_{\min}}$  where  $\lambda_{\min}$  is the smallest eigenvalue of  $H^{-1}(y^0)$  [Grötschel *et al.* 88].

It may seem superfluous to carry out all this effort while having at hand  $y^0$ , the volumetric center, which could work quite well as the test point, but for the sake of completing the discussion and utilizing the movement direction as suggested, we write the new test point as:

$$y^{k+1} = y^k - \alpha \frac{\sum g_i}{\|\sum g_i\|}; \quad \alpha = \sqrt{\lambda_{\min}} \text{ (half-width)} \quad (4.31)$$

Whether one uses the above method for estimating the width or some other method, the long step given by (4.31) may result with a test point which is exterior to the polytope. The effectiveness of taking a step equal to the half-width of the inscribed ellipsoid is related to the *centrality* of the previous test point. It is clear that as one proceeds with the iterations, the subsequent test points may become quite distant from the current centers, and the occasion of infeasibility may occur. One



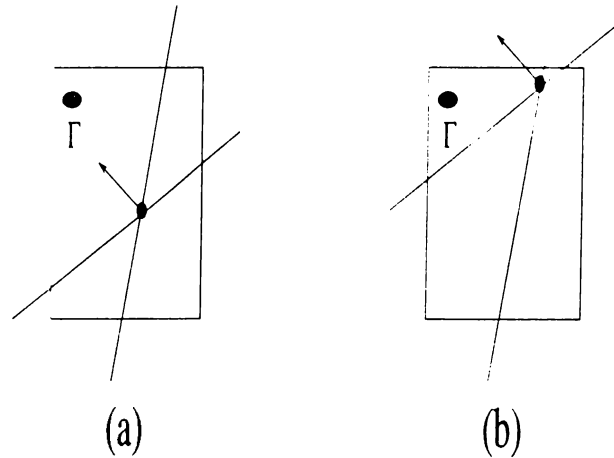


Figure 4.2: (a) A feasible step. (b) An infeasible step.

particular way of restoring feasibility when this occurs is to consider the volumetric center or the analytic center (if available) as the new test point.

So a methodology that can deal with infeasibility is essential. Another advantage of such an approach is the fact that the ability to restore feasibility allows one to take longer steps within an algorithm. Even though one could devise a safe method which stays in the polytope, computational experience indicates that taking long steps are worth considering, since they induce faster termination. It is reported in [Kojima *et al.* 93] that taking longer steps along the central path in primal-dual scaling results in better convergence. So it is worthwhile to estimate a suitable step length and deal with infeasibility.

Numerical experimentation and their theoretical implications will be the subject of future research. Initially, we have found it more promising to investigate shrinkage through rectangular cuts. The ideas underlying this approach are given in the following sections.

Before closing this section we will mention an interesting point related to the NSNG direction. As described, a step is taken in this direction at each iteration. If we had a single function defining the convex body (only an  $f_1 \leq 0$ , such as an ellipsoidal region) the NSNG direction is obtained from (4.31) as  $-(\nabla f_1 / \|\nabla f_1\|)$ . This can be recognized as the steepest descent direction for the nonlinear unconstrained problem 'minimize  $f_1$ '. Recalling that the convex body contains a ball with radius  $\epsilon$  by definition, a convergence criterion similar to that of the conventional steepest descent

algorithm, can be given based on this definition.

#### 4.4 Estimation of the Shrinkage Rate of Containing Domain Procedures

It is established in [Gof. *et al.* 93b] that the algorithm should have terminated whenever the number of iterations  $k$  satisfy:

$$\frac{\epsilon^2}{m} \geq \frac{\frac{1}{2} + 2m \ln(1 + \frac{k+1}{8m^2})}{2m + k + 1} e^{-2\alpha \frac{k+1}{k+1+2m}} \quad (4.32)$$

where  $m$  is the dimension of the Euclidean space in which the convex body is found,  $\epsilon$  the radius of the full dimensional ball contained in the convex body, and  $\alpha$  a positive amount of reduction guaranteed on the potential function. The above statement indicates that the algorithm is a fully polynomial approximation scheme; however it is possibly an overestimation of the average performance. For parameter values of  $m = 3$ ,  $\epsilon = 0.05$ ,  $\alpha = 3.95 \times 10^{-3}$  the minimal value of  $k$  satisfying (4.32) is about 48,500. For typical problems of this size, one does not expect such a large value of  $k$ , on average.

Instead, let us try to track the shrinkage rate of the maximal inscribed ellipsoid—with the restriction of [Vaidya 89]—at each iteration. Assume that we have an algorithm which guarantees that the next ellipsoid has a volume less than a fraction  $\rho$  of the latest ellipsoid, i.e.  $\rho V^k \geq V^{k+1}$ . The volume of the ellipsoid is [Grötschel *et al.* 88]:

$$V^k = \sqrt{\det H^{-1}(y^0)} V_m \quad (4.33)$$

where  $y^0$  is the volumetric center,  $H(y^0)$  is the Hessian of  $-\phi(y)$  (the negated barrier function), and  $V_m = \frac{\pi^{m/2}}{\Gamma(m/2+1)}$  is the volume of the  $m$  dimensional unit ball.

Initially we have a ball of radius  $1/2$  ( $m$  dimensional):

$$V^0 = 2^{-m} V_m \quad (4.34)$$

Given that the convex body contains a ball of radius  $\epsilon$ , the minimal volume that can be achieved just before termination of the algorithm is (after  $k$  iterations):

$$V^k = \epsilon^m V_m \quad (4.35)$$

Since we have assumed that  $\rho$  is the upper bound value of the fraction of volume reduction throughout the algorithm, after  $k$  iterations:

$$\rho^k = \frac{\epsilon^m V_m}{2^{-m} V_m} = (2\epsilon)^m \quad (4.36)$$

$$k = \frac{m \ln(2\epsilon)}{\ln \rho} \quad (4.37)$$

So after  $k = (m \ln(2\epsilon))/(\ln \rho)$  steps, the algorithm should have terminated. With parameter values  $m = 3$ ,  $\epsilon = 0.05$ ,  $\rho = 0.95$  the given algorithm would stop after at most 135 iterations.

It may be theoretically difficult to obtain a worst case value  $\rho$  for some algorithms. Nevertheless, if an average shrinkage rate can be deduced, equations (4.36) and (4.37) are useful to obtain the number of steps required, on average.

## 4.5 An Approach Combining Containing Domains and Descent Directions

The central cutting plane approach to the feasibility problem in [Ye 89], [Gof. *et al.* 93a, b], [Bahn *et al.* 94] yields more and more complex containing domains as the number of iterations grow. For practical implementations one should reconstitute a simpler domain after a prespecified number of iterations by relaxation. Another way to maintain a simple domain is to put certain restrictions on arbitrary cuts. The idea of rectangular cuts has been proposed in [Özak. 95] and developed in [Özak. 96] through a simplification of the central cuts obtained in [Gof. *et al.* 93a, b]. Our objective here is to present a thorough description of this approach.

One can obtain the induced rectangular cuts from a given valid inequality. Each valid inequality provides a lower or upper bound for any variable component which has a nonzero coefficient in the valid inequality. Thus one can possibly obtain  $m$  induced rectangular cuts (since the feasibility problem is defined in  $\Re^m$ ) from each valid inequality. Obviously, some (actually most) of these cuts are redundant.

Thus we will repose the feasibility problem as:

$$\Omega^0 = \{y \in \Re^m : 0 \leq y \leq 1\}$$

$$\Gamma = \{y \in \mathfrak{R}^m : f_i(y) \leq 0\} \quad f_i \text{ -- convex and differentiable}$$

$$\Gamma \subseteq \Omega^k \subseteq \Omega^{k-1} \subseteq \dots \subseteq \Omega^0$$

where  $\Omega^k = \{y \in \mathfrak{R}^m : l^k \leq y \leq u^k\}$  is the current containing polytope  
and  $y^k$  is the current test point in  $\Omega^k$

Hence at the  $k^{\text{th}}$  iteration one has to update a polytope of the form  $\Omega^k = \{y \in \mathfrak{R}^m : l^k \leq y \leq u^k\}$  where  $l^k$  and  $u^k$  are lower and upper bound vectors respectively. It is natural to start with the unit cube as the containing polytope. The current center of the polytope is simply  $y^k = \frac{l^k + u^k}{2}$ . In this approach, no special effort is made for center computations.

One major disadvantage of this approach is the reduced shrinkage rate, when compared to the approach employing arbitrary cuts. Actually, as will be illustrated below, this disadvantage may be so severe that the algorithm might become stuck at a point without a feasible result. However, the simplicity of the containing box enables us to use additional tools to formulate a convergent algorithm. The notion of the negative sum of normalized gradients (NSNG) direction, which has been suggested previously for the cutting plane approach with arbitrary cuts, will be quite useful. By the utilization of this direction we aim to obtain test points which are better than central points.

## 4.6 Obtaining Rectangular Cuts Analytically and the Primitive Rectangular Cutting Plane Algorithm

The method that is described here for obtaining the new containing box induced from the valid cuts, does not necessarily give the minimal box containing the new polytope. However, it is quite simple and straightforward, and one need not bother with the minimal box which is quite difficult to compute. Provided that we have a valid cut  $ay \leq a\bar{y}$  (obtained by a single implementation of the separation oracle, as in [Gof. *et al.* 93a], [Ye 89] etc.), the idea is to obtain the lower or upper bound induced on each component of  $y$  (Figure 4.3). The induced upper (lower) bound is a rectangular cut,  $y_j \leq \hat{u}_j$  ( $\hat{l}_j \leq y_j$ ) (orthogonal to the  $y_h$  axes,  $h = 1, \dots, m$ ,  $h \neq j$ ) and is less deep than  $ay \leq a\bar{y}$ . Hence it is also a valid cut.

Thus each valid cut  $ay \leq a\bar{y}$  induces several valid rectangular cuts (each in the

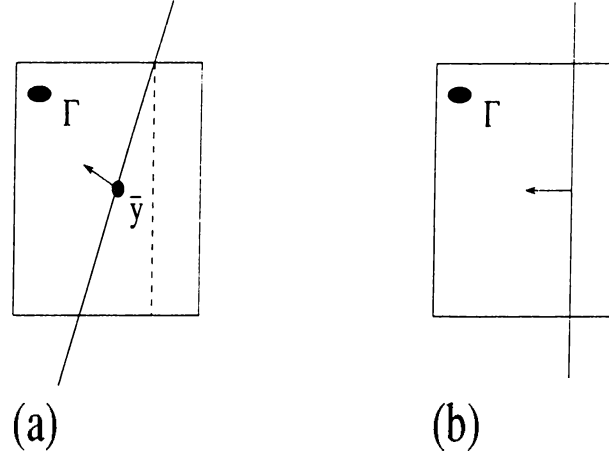


Figure 4.3: (a) The given valid cut,  $ay \leq a\bar{y}$ . (b) The induced rectangular cut,  $y_j \leq \hat{u}_j$ .

form of a lower or an upper bound). These bounds will be determined sequentially for  $j = 1, \dots, m$ , in the following way: Pick a component of  $y$ , say  $y_j$ . If  $a_j > 0$ , then one can obtain a valid upper bound by assigning the lower bound values to the variables with positive coefficients and upper bound values to the variables with negative coefficients, in the valid inequality  $ay \leq a\bar{y}$ . Similarly, if  $a_j < 0$ , then one can obtain a valid lower bound by assigning the lower bound values to the variables with positive coefficients and upper bound values to the variables with negative coefficients. In this way one can obtain a bound induced on  $y_j$  by the valid inequality. (If  $a_j = 0$ , then no bound is induced on  $y_j$ .)

The proposed method of obtaining upper and lower bounds can be verified as follows. When a valid inequality is at hand, one should consider the worst cases, in order to obtain rectangular cuts which are also valid. So, whenever  $a_j > 0$ , by assigning  $l_h \rightarrow y_h$  for  $a_h > 0$  or  $u_h \rightarrow y_h$  for  $a_h < 0$ ,  $h = 1, \dots, m$ ,  $h \neq j$ , we have the induced cut:

$$a_j y_j \leq [a\bar{y} - \sum_{a_h > 0} a_h l_h - \sum_{a_h < 0} a_h u_h] \quad (4.38)$$

For all the worst case possibilities to hold, the RHS should be as large as possible, and this is indeed the case for the above inequality. Similarly, whenever  $a_j < 0$ , the induced cut (by a similar assignment) is:

$$-a_j y_j \geq [-a\bar{y} + \sum_{a_h > 0} a_h l_h + \sum_{a_h < 0} a_h u_h] \quad (4.39)$$

This time, the RHS should be as small as possible for the worst cases and clearly

it turns out to be so when  $y_h = l_h$  if  $a_h > 0$ ,  $y_h = u_h$  if  $a_h < 0$  for  $h = 1, \dots, m$ ,  $h \neq j$ .

Algorithmically, we prefer to treat each valid cut separately and sequentially to obtain the induced rectangular cuts. At the end of an iteration, we update the containing box by the best bounds obtained. Hence the containing box which is obtained at the end of the iteration is not necessarily the minimal box which contains the containing polytope obtained by the introduction of all arbitrary valid cuts (Figure 4.4). But, it is neither meaningful nor easy to compute the minimal box containing that polytope. Before going into the rectangular cutting plane algorithm,

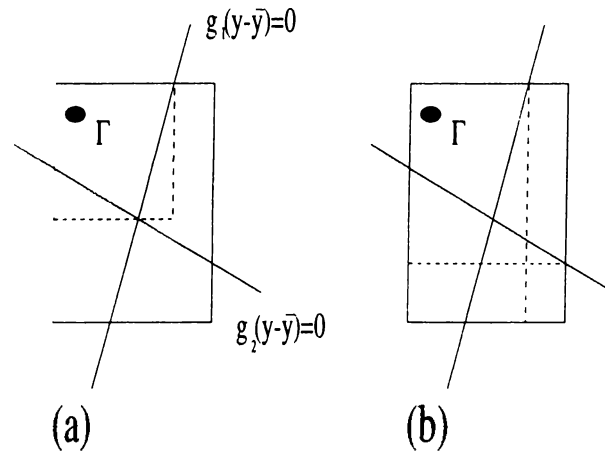


Figure 4.4: (a) The minimal box containing the new polytope found by two valid inequalities. (b) Another containing box computed (as described in this section) which contains the new polytope. This one is not minimal.

specific bounds will be given for the case where the test points are the center points, i.e. at the  $k^{\text{th}}$  iteration  $\bar{y} = \frac{l^k + u^k}{2}$ . Plugging in this value for  $\bar{y}$ , inequalities (4.38) and (4.39) become respectively (from the valid cut  $ay \leq a \frac{l^k + u^k}{2}$ ):

$$\text{(if } a_j > 0), y_j \leq \frac{1}{2}(l_j^k + u_j^k) + \sum_{h=1, h \neq j}^m \left| \frac{a_h}{2a_j} \right| (u_h^k - l_h^k) \quad (4.40)$$

$$\text{(if } a_j < 0), y_j \geq \frac{1}{2}(l_j^k + u_j^k) + \sum_{h=1, h \neq j}^m \left| \frac{a_h}{2a_j} \right| (l_h^k - u_h^k) \quad (4.41)$$

So the modified algorithm restricted to only rectangular cuts (with central test points) will be given as follows:

**The Primitive Rectangular Cutting Plane Algorithm:**

$$\begin{aligned}
& \text{Step 0. } y \in \mathfrak{R}^m, \quad l^0 \leq y \leq u^0, \quad l^0 = \bar{0}, \quad u^0 = e \\
& k = 0, \quad y^0 = \frac{1}{2}e \quad \Omega^0 = \{y : l^0 \leq y \leq u^0\} \\
& \hat{u} = u^0 \quad \Gamma = \{y : f_i(y) \leq 0\} \quad f_i \rightarrow \text{convex and differentiable,} \\
& \hat{l} = l^0 \quad \Gamma \subset \Omega^0
\end{aligned}$$

(We have taken the unit cube as the initial box for convenience, but one may start with any arbitrary box and its center.)

**Step 1.** Check if  $f_i(y^k) < 0$  for all  $i$ . If this is the case, stop with success. If not, for all  $i$  which have  $f_i(y^k) \geq 0$ , consider the valid inequalities of the form:

$$a^i y \leq a^i y^k \quad \text{s.t. } a^i = \nabla f_i(y^k) = g_i, \text{ where } f_i(y^k) \geq 0$$

Here,  $a^i$ 's are not necessarily normalized and if some  $f_i$ 's are not differentiable, subgradients provided by the oracle can be used. For each such cut, update the following upper and lower bounds (rectangular cuts):

For  $j = 1, \dots, m$

(i) if  $a_j > 0$ ; let

$$\tilde{u}_j = \frac{1}{2}(l_j^k + u_j^k) + \sum_{h=1, h \neq j}^m \left| \frac{a_h}{2a_j} \right| (u_h^k - l_h^k)$$

Let  $\hat{u}_j = \min\{\hat{u}_j, \tilde{u}_j\}$

(ii) if  $a_j < 0$ ; let

$$\tilde{l}_j = \frac{1}{2}(l_j^k + u_j^k) + \sum_{h=1, h \neq j}^m \left| \frac{a_h}{2a_j} \right| (l_h^k - u_h^k)$$

Let  $\hat{l}_j = \min\{\hat{l}_j, \tilde{l}_j\}$

**Step 2.**  $u^{k+1} = \hat{u}$ ,  $l^{k+1} = \hat{l}$ ,  $\Omega^{k+1} = \{y : l^{k+1} \leq y \leq u^{k+1}\}$ ,  $y^{k+1} = \frac{l^{k+1} + u^{k+1}}{2}$ . If  $y^{k+1} = y^k$ , then the algorithm fails, otherwise let  $k = k + 1$  and go to **Step 1**.

This algorithm is a nonconvergent routine (it may not terminate with a feasible result), but it illustrates the idea of rectangular cuts for a central cutting plane approach. Actually it will form the basis of a more sophisticated algorithm that will be illustrated in the following section.

Hence we will refer to the stated algorithm as the weak (or the primitive) routine. Indeed, most of the rectangular cuts obtained are redundant, and the nonredundant cuts are not deep. So the shrinkage rate of the containing box is not very impressive and what is more unfortunate is that the algorithm may terminate at an infeasible test point, without a further reduction of the box.

However, the simplicity of the algorithm and the well defined and simple containing domain are still the valuable assets of this approach. As an improvement we will consider the use of noncentral test points which can be possibly better than center points.

In previous sections, the concept of the NSNG direction was introduced. The applicability of this concept was quite limited, since we had been working on arbitrary polytopes without well defined structures, where we had been trying to estimate the polytope radius to obtain a usable step size. Now, since we have a containing domain with a simple and clear definition, we can utilize this concept to its utmost extent. So from now on, we will not consider the pure algorithm given in this section with central test points, but an algorithm which allows movement towards the convex set.

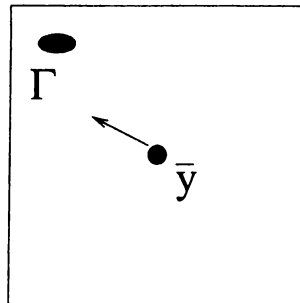


Figure 4.5: Movement in the NSNG direction.

## 4.7 A New Algorithm Based on Containing Boxes and Subgradients

In this section we outline an extended approach to increase the performance of the *weak routine* summarized above. We utilize the NSNG direction described previously,



at each iteration. In this way the algorithm will continue to proceed, even when further shrinkage is no longer possible from a given test point (see Figure 4.6). Furthermore, we expect the generated test points to be better than central points (Figure 4.7).

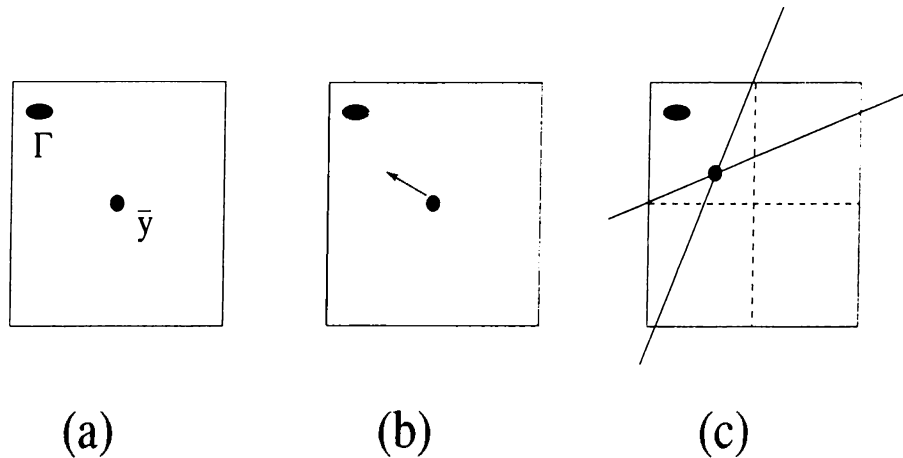


Figure 4.6: (a) Shrinkage does not occur at  $\bar{y}$ . (b) New test point  $y^+ = \bar{y} - \alpha d$ . (c) New containing box computed at the new test point  $y^+$ .

For arbitrary *test and cut* algorithms which call implicit oracles (either to verify feasibility/optimality or to generate valid inequalities), usage of central test points is the most meaningful policy, since this conceptually gives the best shrinkage rate of the containing domain. However, in the specific case where an explicit definition of the convex set is available (through a set of functional inequalities), the predefined movement direction yields better test points than central test points. We still compute the maximum possible step size in a given direction (thus we have to update  $l$  and  $u$ ); however the containing box loses its importance with this approach.

At each iteration we update  $l$  and  $u$  as in the original algorithm. Then we determine the movement direction  $-d$  and a suitable step  $\alpha < \alpha_{\max}$  is taken in this direction. This new point is the test point to be used in the next iteration. As will be seen in later sections, a suitable step sizing subroutine is essential for convergence. At this instant we will assume that the step size can be determined efficiently by a suitable subroutine. We will return to the issue of step sizes later.

### The Extended Rectangular Cutting Plane Algorithm:

**Step 0.**  $y \in \mathfrak{R}^m$ ,  $l^0 \leq y \leq u^0$ , ( $l^0 = \bar{0}$ ,  $u^0 = e$   $y^0 = \frac{1}{2}e$ )

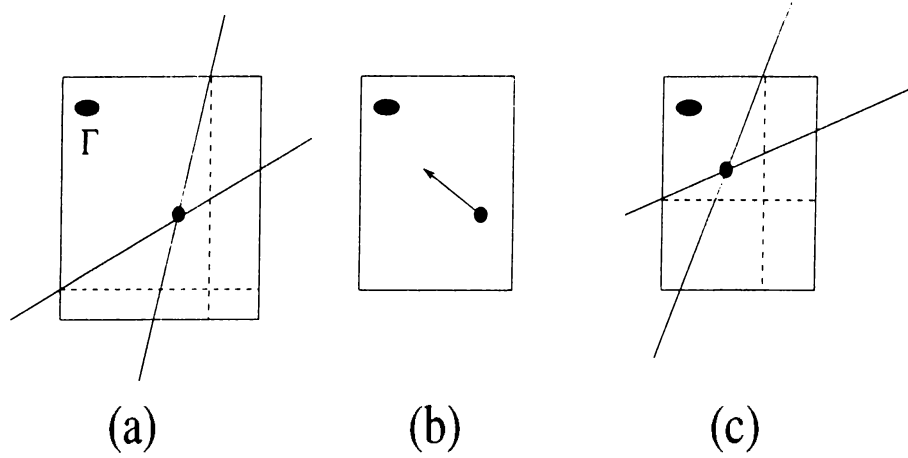


Figure 4.7: (a) Test point  $\bar{y}$ , and shrinkage through rectangular cuts. (b) New test point  $y^+ = \bar{y} - \alpha d$ , obtained by moving in the direction of  $-d$ . (c) New test point  $y^+$ , and shrinkage afterwards.

$$k = 0, \quad \hat{u} = u^0, \quad \hat{l} = l^0, \quad \Omega^0 = \{y : l^0 \leq y \leq u^0\}$$

$$\Gamma = \{y : f_i(y) \leq 0\} \quad (\text{All } f_i \text{'s are convex and differentiable})$$

$$\Gamma \subset \Omega^0$$

**Step 1.** Check for all  $i$ , whether  $f_i(y^k) < 0$ . If so, then stop; an interior point has been found. If not, then for all  $i$  such that  $f_i(y^k) \geq 0$ , consider the following valid inequalities:

$$a^i y \leq a^i y^k \quad \text{s.t. } a^i = \frac{g_i}{\|g_i\|}, \quad \text{where } f_i(y^k) \geq 0$$

Here  $g_i = \nabla f_i(y^k)$  and  $a_i$ 's are normalized. For each such cut, update the following upper and lower bounds:

For  $j = 1, \dots, m$

(i) if  $a_j > 0$ ; let

$$\tilde{u}_j = y_j^k + \sum_{h:a_h>0} \frac{a_h}{a_j} (y_h^k - l_h^k) + \sum_{h:a_h<0} \frac{a_h}{a_j} (y_h^k - u_h^k)$$

and let  $\hat{u}_j = \min\{\hat{u}_j, \tilde{u}_j\}$

(ii) if  $a_j < 0$ ; let

$$\tilde{l}_j = y_j^k - \sum_{h:a_h>0} \frac{a_h}{a_j}(y_h^k - l_h^k) - \sum_{h:a_h<0} \frac{a_h}{a_j}(y_h^k - u_h^k)$$

and let  $\hat{l}_j = \max\{\hat{l}_j, \tilde{l}_j\}$

**Step 2.** Let

$$-d = -\frac{\sum_{i:f_i(y^k) \geq 0} a^i}{\left\| \sum_{i:f_i(y^k) \geq 0} a^i \right\|}$$

Let  $u^{k+1} = \hat{u}$ ,  $l^{k+1} = \hat{l}$  and  $y^k = y^k - \alpha d$  for a suitable step size  $\alpha$ . Set  $k = k + 1$  and go to **Step 1**.

## 4.8 Some Concepts Related to the Descent Approach

We will classify the extended version of the rectangular cutting plane algorithm as a descent approach, since iterationally the algorithm takes a step in a mixed descent direction towards the feasible convex body. The term *descent* is interpreted as an overall reduction of the individual functional values, where the functional inequalities define the convex body.

It was emphasized previously, that the step sizing policy is of central importance in determining the convergence behavior of the algorithm. One can adopt two different approaches to step sizing. The first approach is to take step sizes as predefined/updated fractions of the maximum possible step size that will keep us in the containing domain (such as  $0.7 \times \alpha_{\max}$ ,  $0.5 \times \alpha_{\max}$ ,  $0.2 \times \alpha_{\max}$ ). The second approach is to use step sizes unrelated to the maximum possible step. In this case, the containing box is no longer of any importance and the routine becomes a pure descent algorithm. In this approach, one has to use reference measures other than those of the containing box, for determining the step size at each iteration.

A number of definitions will be given before we go into theoretical results:

### The Descent Set:

A movement direction from a test point will be denoted as an *overall descent direction* if a positive step towards that direction yields a reduction in all nonnegative

functional values. Taking steps in such consecutive descent directions would theoretically provide a convergent algorithm and this fact makes the concept important. The descent set, with respect to the test point  $\bar{y}$  will be defined as the set of points (convex) where each point has a reduced value of the functional values which are nonnegative at  $\bar{y}$ .

Obviously the descent set contains the convex body, since every point  $y$  within that set satisfies  $f_i(y) \leq 0$  for all  $i$  s.t.  $f_i(\bar{y}) \geq 0$ . Actually the descent set can be defined as the intersection of convex sets bounded by the contour sets of all nonnegative functions passing through the test point  $\bar{y}$ . In more formal terms:

$$\bigcap_{i: f_i(\bar{y}) \geq 0} \{f_i(y) \leq f_i(\bar{y})\} \quad (4.42)$$

Naturally the set is convex. Any direction vector emanating from the test point which traverses the descent set (which has common points with the descent set) is an overall descent direction at this test point.

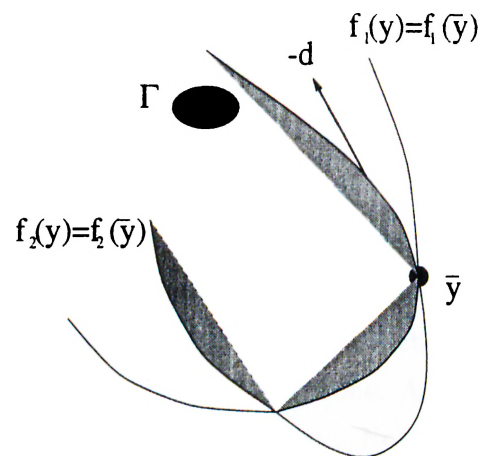


Figure 4.8: The descent set and a descent direction,  $-d$ .

### The Descent Cone:

Now we introduce a slightly different concept in order to obtain a better definition for the descent direction. Considering only those functions with nonnegative values at the test point, it will be recalled that each functional inequality induces a valid cut of the form  $g_i y \leq g_i \bar{y}$ , assuming that all  $g_i$ 's are normalized. We will name the intersection of these halfspaces as the descent cone. The cone is pointed at the test

point  $\bar{y}$ . Formally the cone is defined as:

$$\bigcap_{i: f_i(\bar{y}) \geq 0} \{g_i y \leq g_i \bar{y}\} \quad (4.43)$$

The hyperplane  $g_i y = g_i \bar{y}$  is tangent to the contour set  $f_i(y) = f_i(\bar{y})$  and clearly the steepest descent direction  $g_i$  is perpendicular to this tangent hyperplane.

Now let  $y^+ = \bar{y} - \alpha d$  for some  $\alpha > 0$ . We can say that  $-d$  is a descent direction if and only if  $y^+$  is contained in the descent cone, defined as in (4.43). This is expected because  $y^+$  as in (4.42) should satisfy  $f_i(y^+) \leq f_i(\bar{y})$  for all  $i$  with  $f_i(\bar{y}) \geq 0$ , for some positive value of  $\alpha$ . The reverse statement is also evident: if any  $y^+$  was outside the descent cone, this would indicate that at least one inequality of the form  $f_i(y^+) > f_i(\bar{y})$  holds; which would also violate the descent condition.

The descent set is contained in the descent cone. This is because each hyperplane in (4.43) is a supporting hyperplane of the descent set at the test point  $\bar{y}$ . Furthermore, it is easy to verify that the inequality  $dy \leq d\bar{y}$  is valid for the convex body (see the following section), where  $d$  is defined as (from now on we relieve the assumption of the  $g_i$ 's being normalized):

$$d = \frac{\sum_{i: f_i(\bar{y}) \geq 0} \frac{g_i}{\|g_i\|}}{\|\sum_{i: f_i(\bar{y}) \geq 0} \frac{g_i}{\|g_i\|}\|} \quad (4.44)$$

We will refer to this surrogate hyperplane  $\{y : dy = d\bar{y}\}$  as the *averaged support plane*. Clearly it is orthogonal to the movement direction  $-d$  and it is a supporting hyperplane for the descent set and the descent cone. Another support plane can also be given as  $Gy \leq G\bar{y}$ , where  $G = \sum_{i: f_i(\bar{y}) \geq 0} g_i$ . Note that  $G$  is not identical with  $d$ . In fact,  $d = \sum_{i: f_i(\bar{y}) \geq 0} \frac{g_i}{\|g_i\|}$ , is a scaled version of  $G$ .

**Remark:** If the smallest angle between the conic rays of the descent cone is greater than 90 degrees, then  $-d$  (the NSNG direction) is a descent direction.

### The Ascent Cone:

In a similar manner we can define the ascent cone, which is also pointed at  $\bar{y}$ . Similarly, it is the intersection of several half spaces:

$$\bigcap_{i: f_i(\bar{y}) \geq 0} \{g_i y \geq g_i \bar{y}\} \quad (4.45)$$

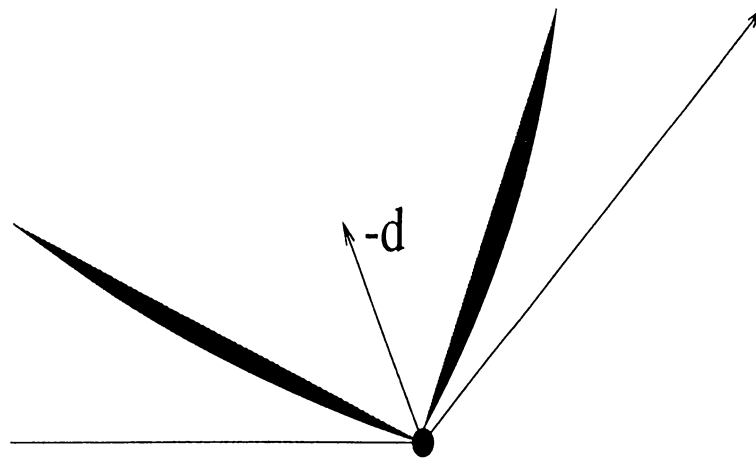


Figure 4.9: The descent cone and a descent direction  $-d$ .

It can be seen that the only difference is that the inequalities in (4.43) have changed their direction. As the name implies, if  $y^+ = \bar{y} - \alpha d, (\alpha > 0)$  is contained in the ascent cone, then we should have  $f_i(y^+) \geq f_i(\bar{y})$  for all  $i$  with  $f_i(\bar{y}) \geq 0$ . This fact follows from the convexity of these functions.

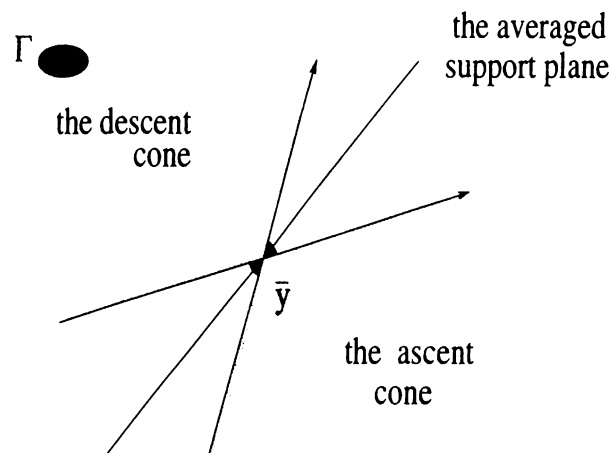


Figure 4.10: The descent cone, the ascent cone, the averaged support plane and a descent direction  $-d$ .

**The Functional Distance:**

It has been stated previously that, if the step lengths are short enough, then the algorithm should converge. Experimental evidence indicates that if the step lengths are not short enough, then it is quite possible that the algorithm may not terminate. This phenomenon takes place due to oversized steps, in the form of orbiting around

the convex set without reaching it.

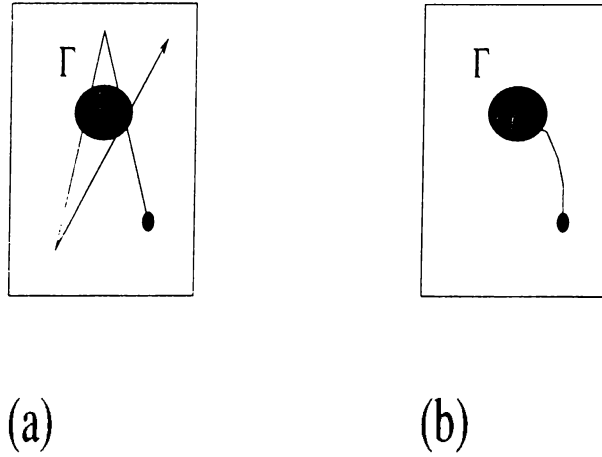


Figure 4.11: (a) Long steps. (b) Short steps.

What should be the step size, so that orbiting does not occur? Intuitively, if the consecutive step sizes are longer than the Euclidean distance between the test point and the convex set, then this will result in an orbital trajectory.

However, determining the distance to the convex set could be quite difficult in practice. (In fact, if we knew this distance, we could almost simplify the convex feasibility problem to a linear feasibility problem.) For this reason, we will introduce the following relative distance measure instead of the Euclidean distance. The functional distance will be defined as:

$$F(\bar{y}) = \sum_{i: f_i(\bar{y}) \geq 0} f_i(\bar{y}) \quad (4.46)$$

We will also refer to  $F$  as the functional sum, since it is the sum of the positive functional values. In the next section, it will be verified that it is a convex function.

Despite its simple definition, the functional distance is a useful surrogate for the Euclidean distance, allowing relative comparisons between consecutive algorithmic steps. If it decreases continually, then this means that we have an overall descent behavior. If it fluctuates between different values or if it almost remains constant, then we can say that the algorithm is trapped in an orbit, hence the current step sizes are not short enough, and they must be adjusted. In this way, we do not need to have an explicit knowledge of the Euclidean distance to judge whether the step lengths are longer than necessary.

Due to convexity, as one moves away from the convex set in a fixed direction, the functional distance will also increase. However, the rate of increase in different directions will be quite different, since at a certain test point, the functional values are quite different. Furthermore, the functions independently have irregular increase rates for a fixed direction. There is a complicated relationship between the functional distance and the distance to the convex set. It is not our objective to establish such a relationship.

For practical purposes we will define the functional scaled distance (sum) as:

$$F_{Sc}(\bar{y}) = \sum_{i: f_i(\bar{y}) \geq 0} \beta_i f_i(\bar{y}) \quad (\beta_i > 0) \quad (4.47)$$

The scaled sum is also convex.

We need one more definition, which we will refer to as the weighted functional sum:

$$F_w(\bar{y}) = \sum_{i: f_i(\bar{y}) \geq 0} \frac{f_i(\bar{y})}{\|g_i\|} \quad (4.48)$$

Clearly  $F_w$  is a special case of  $F_{Sc}$ , with  $\forall i, \beta_i = \frac{1}{\|g_i\|}$ . While establishing convergence of the algorithm in the following section, we will mostly make use of the weighted functional distance.

### An Overall Reduction Algorithm:

Finally, we define the concept of *overall reduction* of a certain function. Let the initial value of the function be finite and positive. Consider some primal minimization algorithm. At any given iteration  $k$ , if there exists a positive finite integer  $N$  such that  $f(y^k) > f(y^{k+N})$  (meaning that after  $N$  iterations a positive amount of reduction is guaranteed on  $f$ ), then we will denote this algorithm as an overall reduction algorithm on  $f$ .

Note that  $N$  need not be the smallest integer satisfying  $f(y^k) > f(y^{k+N})$ . Moreover, for an overall reduction algorithm, there can be a finite or infinite number of reduction steps depending on the convergence routine.



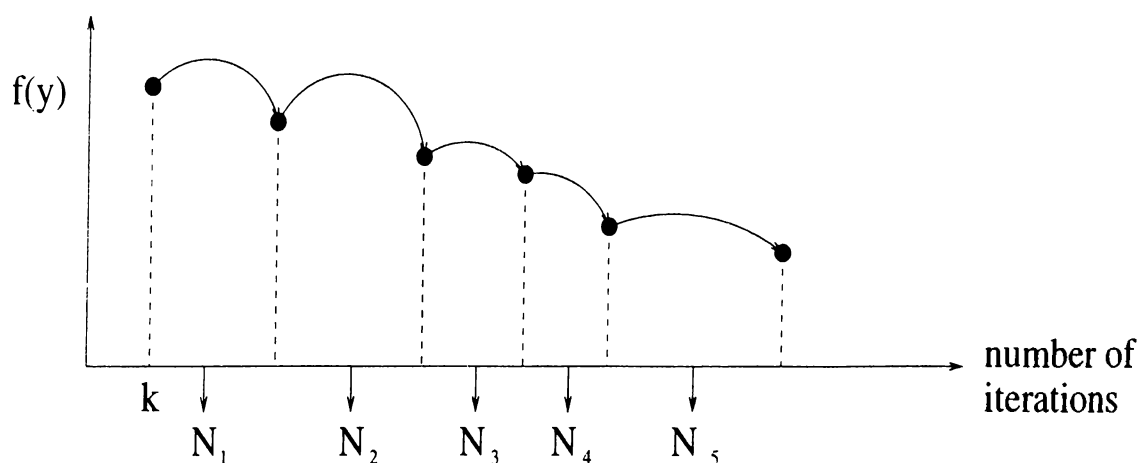


Figure 4.12: An overall reduction process of a certain function.

## 4.9 Convergence Results

In the remaining part of this chapter, we will mainly deal with the extended cutting plane approach. It has been stated earlier that the convergence behavior is governed by the step sizing policy. We will give the theoretical convergence of the extended cutting plane routine provided that an adaptive step sizing policy is used.

Experimental results indicate that if the step sizes are small enough, then the algorithm should converge. Therefore, the critical issue is to determine ‘how small is small enough’ for convergence. We will see that by keeping track of the test points, it is possible to properly adjust the step size.

At this point, one remark will be made regarding the NSNG direction, which was defined as:

$$-d = -\frac{\sum_{i:f_i(\bar{y}) \geq 0} \frac{g_i}{\|g_i\|}}{\|\sum_{i:f_i(\bar{y}) \geq 0} \frac{g_i}{\|g_i\|}\|} \quad (4.49)$$

That is, at a given test point  $\bar{y}$ , we sum up all the normalized gradients of all the functions which have nonnegative values. Steps are taken in the direction of  $-d$ .

Suppose that we have only a single function  $f_i$  which has a nonnegative value at point  $\bar{y}$ . Then, the movement direction given by (4.49) is the steepest descent direction for the problem ‘minimize  $f_i$ ’. In fact, the algorithm minimizes in a certain way several convex functions by taking a combined (steepest descent) direction, and stops when all functions have negative values. This similarity to the steepest descent

algorithm will be essential in establishing the convergence criteria of the algorithm.

### 4.9.1 Some Lemmas Related to Convergence

In this section we will list a number of elementary lemmas which are of importance for establishing convergence. We assume that all functions underlying the convex set are explicitly known, convex and differentiable. The assumption of differentiability will be relaxed later whenever generalizations are possible but unless stated so, we will retain this assumption for convenience.

Due to convexity of the functions  $f_i$ ,  $i = 1, 2, \dots$  the sets defined as  $\{y : f_i(y) \leq \text{val}_i\}$  are convex. We also sometimes refer to the contour sets of the functions with the prespecified value  $\text{val}_i$  as  $\{y : f_i(y) = \text{val}_i\}$ . Naturally these contour sets form the boundaries of their corresponding convex sets. The convex body for which we seek an interior point is  $\Gamma = \bigcap_i \{y : f_i(y) \leq 0\}$ .

**Lemma 4.9.1** For all  $i$ ,  $\text{val}_1 < \text{val}_2 \Leftrightarrow \{y : f_i(y) \leq \text{val}_1\} \subseteq \{y : f_i(y) \leq \text{val}_2\}$  given  $\text{val}_1, \text{val}_2 \geq \min\{f_i(y)\}$ .

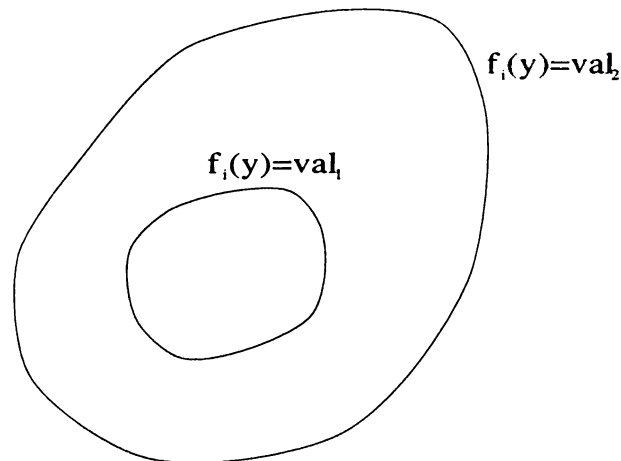


Figure 4.13:  $\text{val}_1 < \text{val}_2 \Leftrightarrow \{y : f_i(y) \leq \text{val}_1\} \subseteq \{y : f_i(y) \leq \text{val}_2\}$

**Proof:** The lemma follows from convexity. ■

**Lemma 4.9.2** For any  $i$ , provided that  $\text{val}_i > 0$   
 $\{y : f_i(y) \leq \text{val}_i\} \supseteq \{y : f_i(y) \leq 0\} \supseteq \Gamma$ .

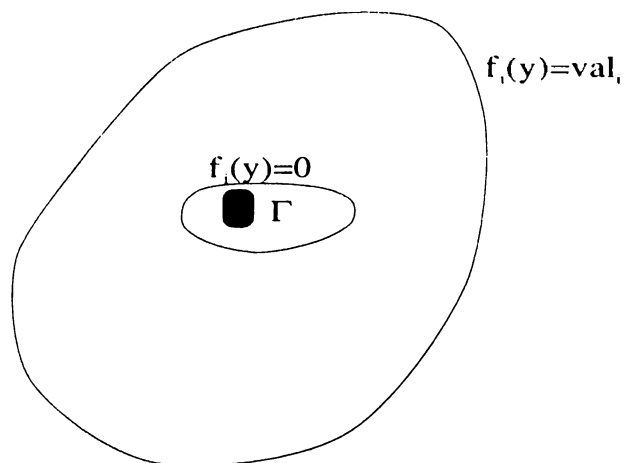


Figure 4.14:  $\{y : f_i(y) \leq \text{val}_i\} \supseteq \{y : f_i(y) \leq 0\} \supseteq \Gamma$ .

**Proof:** This lemma follows from **Lemma 4.9.1** and the definition of  $\Gamma$ . ■

**Lemma 4.9.3** Consider the test point  $\bar{y}$ , for which possibly  $\exists i$  s.t.  $f_i(\bar{y}) \geq 0$ . The convex body is contained in the descent set due to  $\bar{y}$ . Clearly the descent set is nonempty.

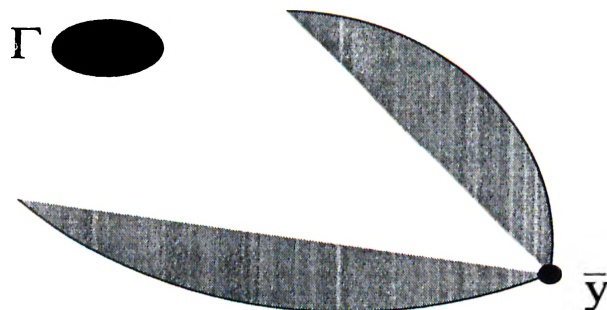


Figure 4.15: The descent set is nonempty and it contains  $\Gamma$ .

**Proof:** The claim is valid since  $\{y : f_i(y) \leq \text{val}_i\} \supseteq \{y : f_i(y) \leq 0\}$  for all  $i$ .

$$\underbrace{\bigcap_{i: f_i(y) \geq 0} \{y : f_i(y) \leq f_i(\bar{y})\}}_{\text{The Descent Set}} \supseteq \bigcap_i \{y : f_i(y) \leq \text{val}_i\} \supseteq \{y : f_i(y) \leq 0\} = \Gamma.$$

■

**Lemma 4.9.4** Consider the test point  $\bar{y}$ , possibly with  $\exists i$  s.t.  $f_i(\bar{y}) \geq 0$ . The averaged support plane defined previously as  $dy \leq d\bar{y}$  is valid for  $\Gamma$ .

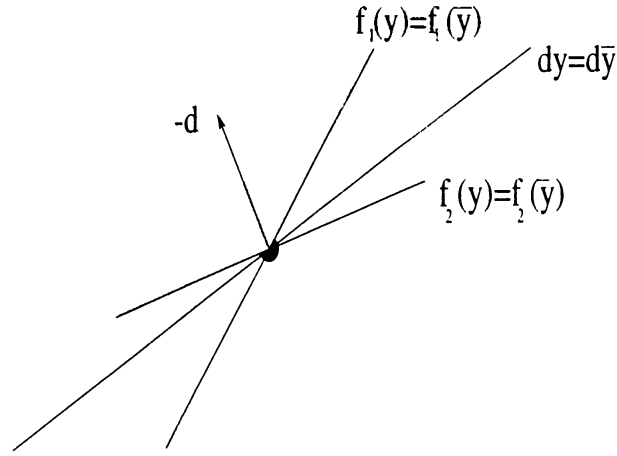


Figure 4.16: The averaged support plane is valid.

**Proof:** The claim follows as a consequence of the following reasoning: For all  $i$ ,  $g_i(y) \leq g_i\bar{y}$  is valid.

Thus,  $\sum g_i(y) \leq \sum g_i(\bar{y})$

Finally,  $Gy \leq G\bar{y}$ , where  $G = \sum g_i$

$$\frac{\sum g_i(y)}{\|\sum g_i\|} \leq \frac{\sum g_i(\bar{y})}{\|\sum g_i\|}$$

Furthermore, any convex combination also gives a valid inequality:  $\sum \beta_i g_i(y) \leq \sum \beta_i g_i(\bar{y})$ ,  $\forall \beta_i \geq 0$

This simply means that the inequality  $dy \leq d\bar{y}$  is valid. ■

**Lemma 4.9.5**  $F(y) = \sum_{i:f_i(y) \geq 0} f_i(y)$  is convex throughout the containing polytope.

**Proof:** To verify, let us define the following function, for all  $i$ :

$$\hat{f}_i(y) = \begin{cases} f_i(y) & \text{if } f_i(y) \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.50)$$

Since  $f_i(y)$  is convex over  $\mathfrak{R}^m$ ,  $\hat{f}_i(y)$  is also convex. With this new definition, we can define:

$$F(y) = \sum_i \hat{f}_i(y) \quad (4.51)$$

Hence  $F$  is convex. ■

**Lemma 4.9.6**  $F_{\text{Sc}}(y) = \sum_{i:f_i(y) \geq 0} \beta_i f_i(y)$  ( $\beta_i > 0$ ). *is also convex.*

**Proof:** The verification is similar, one just needs to define  $\hat{f}_i(y)$  as in (4.47) and redefine:

$$F_{\text{Sc}}(y) = \sum_i \beta_i \hat{f}_i(y) \quad (4.52)$$

■

As a corollary to this claim, we can say that the weighted functional distance as defined in (4.48) is also convex.

In both of these claims we use the basic fact that, a conic combination of convex functions is also convex (see for example [Luen. 84]). Both functions are convex but not differentiable everywhere, throughout the containing polytope.

Note that  $F_w$  has a minimum value of zero. In fact, when such a point is detected, the feasibility problem is solved and even if the point is not in the interior, an interior point can be found by a single movement.

Unfortunately, the movement direction that we have adopted is not necessarily an overall descent direction, despite the fact that it provides a general descent, and that the points obtained are definitely not in the ascent cone. Certain constructed examples indicate that for irregular cases or unbalanced scalings the NSNG directions might fall outside the descent cones. However, since the descent cone pointed at an infeasible point is nonempty, such a direction always exists with a certain scaling as:  $-d = -\sum \beta_i g_i$ ,  $\beta_i \geq 0$ ,  $\sum \beta_i = 1$  where all  $g_i$ 's are normalized. Theoretically, the availability of a descent direction might contribute to the convergence analysis of algorithms which use scaled NSNG directions with adaptive step sizing.

Nevertheless, we postpone the scaling issue to the implementation stage, and pursue our development with the unscaled NSNG direction. It will be seen in the next section that the movement direction is a descent direction for the weighted functional distance, and guarantees convergence through the use of an appropriate step sizing policy. A different approach to establish the convergence of the algorithm will also be sketched in the next section.

### 4.9.2 Establishing Convergence of the Extended Version of the Rectangular Cutting Plane Approach

In this section we will adopt the weighted functional distance given in (4.48) as the reference function, since it has both theoretical and practical significance. It will be seen that the NSNG direction is a descent direction for the weighted functional distance. Actually, for points where  $F_w$  is differentiable, it is indeed the steepest descent direction.

Let us consider the feasibility problem. Initially  $F_w(y^0) = \sum_{i: f_i(y^0) \geq 0} f_i(y^0)$  has a finite value, since it is the sum of a finite number of finite values. By some descent routine over  $F_w$ , the problem will be solved (or almost solved) when  $F_w(y)$  becomes zero. The overall convergence behavior is quite similar to that of a steepest descent routine for the  $\min F_w(y)$  problem. The routine proceeds as  $F_w(y) \rightarrow 0$ , and upon termination one would have  $F_w(\bar{y}) = 0$ , meaning that  $\forall i, f_i(\bar{y}) \leq 0$ ,  $\bar{y}$  satisfies the feasibility problem.  $\bar{y}$  may not be an interior point of  $\Gamma$ , but we can easily find an interior point by a single positive step in the direction of  $-d = -\sum_{i: f_i(\bar{y})} f_i(\bar{y})$ .

Of course, searching the feasible set  $\Gamma$  through steepest descent steps of  $F_w(y)$  does not yield the best path towards  $\Gamma$ . However, this method exhibits a certain simplicity, and furthermore, keeping track of  $F_w$  gives some insight regarding the state of the current test point.

We wish to show first that the NSNG direction is a descent direction for  $F_w$ . It is quite easy to do this at a point where  $F_w$  is differentiable. We will do this in **Theorem 4.9.1**. Proof for the nondifferentiable case is not trivial and will be established through **Lemmas 4.9.7, 4.9.8, 4.9.9** and **Theorem 4.9.2**. We should recall that, we assume that all  $f_i$ 's are differentiable and hence  $g_i = \nabla f_i(y)$ . From (4.48),  $F_w(y) = \sum_{i: f_i(y) \geq 0} \frac{f_i(y)}{\|g_i\|}$ , and we define  $G_w = \sum_{i: f_i(y) \geq 0} \frac{g_i}{\|g_i\|}$ . (Actually  $d$ ,

defined in (4.49), is a normalized redefinition of  $G_w$ .

**Theorem 4.9.1** *Let  $F_w$  be differentiable at some point  $\bar{y}$ . Then  $\nabla F_w(\bar{y}) = \sum_{i:f_i(\bar{y}) \geq 0} \frac{g_i}{\|g_i\|}$ .*

**Proof:** The proposition is true, since  $\nabla F_w(\bar{y}) = \nabla \left( \sum_{i:f_i(\bar{y}) \geq 0} \frac{f_i(\bar{y})}{\|g_i\|} \right) = \sum_{i:f_i(\bar{y}) \geq 0} \left( \frac{\nabla f_i(\bar{y})}{\|g_i\|} \right) = \sum_{i:f_i(\bar{y}) \geq 0} \frac{g_i}{\|g_i\|} = G_w = d$

Hence  $-d$  is the steepest descent direction at some point  $y$  for  $F_w$ . ■

**Lemma 4.9.7** *Let  $\bar{y}$  be an interior point of  $\Omega^k$  so that  $f_i(\bar{y}) \geq 0$  for some  $i \in I_1$  and  $f_i(\bar{y}) < 0$  for some  $i \in I_2$ . Then, there exists a nonempty neighborhood of  $\bar{y}$ ,  $N(\bar{y})$  so that all  $y \in N(\bar{y})$  would satisfy:  $f_i(y) < 0$ , for  $i \in I_2$*

**Proof:** In relation to this neighborhood, for any  $y \in N(\bar{y})$  we will consider the following partition of  $I_1$ :

$$i \in I_{1A} \text{ if } f_i(y) \geq 0$$

$$i \in I_{1B} \text{ if } f_i(y) < 0$$

where  $I_{1A} \cup I_{1B} = I_1$ , is dependent on  $y$ .

One can define a neighborhood such that  $\forall i \in I_2, f_i(y) < 0$ , since for  $i \in I_2$ ,  $f_i(\bar{y})$  is strictly smaller than zero. The partition stated above for  $I_1$  is also trivial. ■

**Lemma 4.9.8**  *$G_w$  is a subgradient for the nondifferentiable point  $\bar{y}$  of  $F_w$  if and only if  $\bar{y}$  solves the problem:*

$$\sup_{y \in \mathbb{R}^m} G_w y - F_w(y) \tag{4.53}$$

or equivalently  $\bar{y}$  solves:

$$\begin{aligned} \max \quad & G_w y \\ \text{s.t.} \quad & F_w(y) \leq F_w(\bar{y}) \end{aligned} \tag{4.54}$$

**Proof:** The problem given in (4.53) is the conjugate function of  $F_w$ . In other words, it is equivalent to the solution of  $F_w^*(G_w)$ . From elementary convex analysis, a necessary and sufficient condition for  $G_w$  to be a subgradient of  $F_w$  is that  $\bar{y}$  be a maximizer of (4.53) [Mc Corm. 83]. If this is the case, then the hyperplane  $G_w(y) = G_w(\bar{y})$  is a support plane of the convex set  $F_w(y) \leq F_w(\bar{y})$ . Hence the linear function  $G_w y$  is maximized over this convex set at the extreme point  $\bar{y}$  (referring to the support plane definition, for example in [Schr. 86]) and thus (4.53) and (4.54) are identical. ■

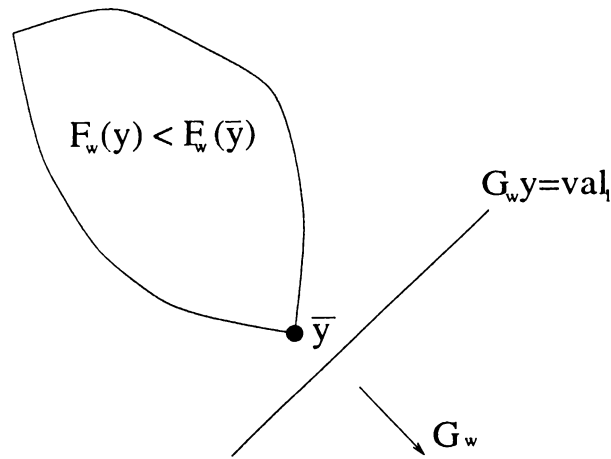


Figure 4.17:  $G_w y$  is maximized over the convex set  $F_w(y) \leq F_w(\bar{y})$  at  $\bar{y}$ .

**Lemma 4.9.9** *A local optimum to problems (4.53) and (4.54) is also a global optimum for both.*

**Proof:** This lemma follows from the facts that (4.54) is a convex minimization problem over a convex set and (4.53) and (4.54) are equivalent. ■

**Theorem 4.9.2** *If  $F_w$  is not differentiable at the test point  $\bar{y}$ , then  $G_w = \sum_{i \in I_1} \frac{g_i}{\|g_i\|}$  where  $g_i = \nabla f_i(\bar{y})$ ,  $I_1 = \{i : f_i(\bar{y}) \geq 0\}$  is a subgradient for  $F_w$  at  $\bar{y}$ .*

**Proof:** We will be looking at two different cases:

Case 1: There exists a neighborhood  $N(\bar{y})$  of  $\bar{y}$  such that,  $y \in N(\bar{y})$



$$\text{satisfying } \begin{cases} f_i(y) \geq 0 & \text{for all } i \in I_1 \\ f_i(y) < 0 & \text{for all } i \in I_2 \end{cases}$$

Within this neighborhood:

$$\begin{aligned} \sup_y G_w y - F_w(y) &= \sup_y \left[ \left( \sum_{i \in I_1} \frac{g_i}{\|g_i\|} \right) y - \sum_{i \in I_1} \frac{f_i(y)}{\|g_i\|} \right] \\ &= \sup_y \left[ \sum_{i \in I_1} \left( \frac{g_i y - f_i(y)}{\|g_i\|} \right) \right] \\ &= \sum_{i \in I_1} \left[ \sup_y \left( \frac{g_i y}{\|g_i\|} - \frac{f_i(y)}{\|g_i\|} \right) \right] \end{aligned}$$

$$\text{Since } g_i = \nabla f_i(\bar{y}), \quad \sup_y \left( \frac{g_i y}{\|g_i\|} - \frac{f_i(y)}{\|g_i\|} \right) = \frac{g_i \bar{y}}{\|g_i\|} - \frac{f_i(\bar{y})}{\|g_i\|}$$

So we have,

$$= \sum_{i \in I_1} \left( \frac{g_i \bar{y}}{\|g_i\|} - \frac{f_i(\bar{y})}{\|g_i\|} \right)$$

$$\text{and } = G_w \bar{y} - F_w(\bar{y})$$

Thus  $\bar{y}$  is a local optimizer of problem (4.53).

Case 1 is a special case of **Lemma 4.9.7** where  $I_{1B} = \emptyset$ . So we will proceed with Case 2 where  $I_{1B}$  is nonempty.

Case 2: In the neighborhood of  $\bar{y}$ , where  $\forall y \in N(\bar{y})$  for all  $i \in I_2$ , any  $y \in N(\bar{y})$

$$\text{satisfies } \begin{cases} f_i(y) < 0 & \text{for all } i \in I_2 \\ f_i(y) \geq 0 & \text{for all } i \in I_{1A} \\ f_i(y) < 0 & \text{for all } i \in I_{1B} \end{cases}$$

where  $I_{1A} \cup I_{1B}$  and  $I_{1B} \neq \emptyset$ .

So if this is the case for any  $y \in N(\bar{y})$  we have:

$$\begin{aligned} & G_w y - F_w(y) \\ &= \sum_{i \in I_1} \frac{g_i}{\|g_i\|} y - \sum_{i \in I_{1A}} \frac{f_i(y)}{\|g_i\|} \\ &= \sum_{i \in I_{1B}} \frac{g_i}{\|g_i\|} y + \sum_{i \in I_{1A}} \frac{g_i}{\|g_i\|} y - \sum_{i \in I_{1A}} \frac{f_i(y)}{\|g_i\|} \\ & \quad \text{Recalling that } \sum_{i \in I_{1B}} \frac{f_i(y)}{\|g_i\|} < 0 \end{aligned}$$

$$\begin{aligned}
&= \sum_{i \in I_{1B}} \frac{g_i}{\|g_i\|} y + \sum_{i \in I_{1A}} \frac{g_i}{\|g_i\|} y - \sum_{i \in I_{1A}} \frac{f_i(y)}{\|g_i\|} < && \sum_{i \in I_{1B}} \frac{g_i}{\|g_i\|} y - \sum_{i \in I_{1B}} \frac{f_i(y)}{\|g_i\|} \\
&&& + \sum_{i \in I_{1A}} \frac{g_i}{\|g_i\|} y - \sum_{i \in I_{1A}} \frac{f_i(y)}{\|g_i\|} \\
&= \sum_{i \in I_{1B}} \frac{(g_i y - f_i(y))}{\|g_i\|} + \sum_{i \in I_{1A}} \frac{(g_i y - f_i(y))}{\|g_i\|} \\
&= \sum_{i \in I_1} \left( \frac{g_i y - f_i(y)}{\|g_i\|} \right) \\
&\leq \sum_{i \in I_1} \left( \frac{g_i \bar{y} - f_i(\bar{y})}{\|g_i\|} \right)
\end{aligned}$$

where the last inequality follows from:

$$\sup_y \left( \frac{g_i y - f_i(y)}{\|g_i\|} \right) = \frac{g_i \bar{y} - f_i(\bar{y})}{\|g_i\|} \quad \text{for all } i \in I_1$$

Hence,  $G_w y - F_w(y) < G_w \bar{y} - F_w(\bar{y})$  for any  $y \in \mathcal{N}(\bar{y})$  so  $\bar{y}$  is a local optimizer for the problem given in (4.53).

Thus for both cases  $\bar{y}$  is a local maximizer of problem (4.53). From **Lemma 4.9.9**  $\bar{y}$  is also a global optimizer. So from **Lemma 4.9.8**  $G_w$  is a subgradient of  $F_w$  for the nondifferentiable point  $\bar{y}$ . ■

**Remark:** In the above proof, one can reduce Case 1 to Case 2 by removing the necessity that  $I_{1B}$  is nonempty. Since it can also be empty—to account for Case 1—we would have  $\sum_{i \in I_{1B}} f_i(y) \leq 0$  so that the strict inequality should be replaced by *less than or equal to* and in the end we would have  $G_w y - F_w(y) \leq G_w \bar{y} - F_w(\bar{y})$ . However, treating the two cases separately provides some insight about the local optimum. For Case 2  $\bar{y}$  is strictly local optimal.

**Theorem 4.9.3** *The extended rectangular cutting plane algorithm converges to an interior point of the feasible set when adaptive step sizing policy is used.*

**Proof:** At each iteration, since  $-d$  is a descent direction for  $F_w$ , there exists an  $\alpha > 0$  such that  $F_w(\bar{y} - \alpha d) < F_w(\bar{y})$ . Hence a positive reduction on  $F_w$  is guaranteed iterationally. Since the initial value of  $F_w$  is assumed to be finite, after a number of iterations, the algorithm should eventually yield  $F_w(\bar{y}) = 0$ . ■

We have stated that an algorithm which utilizes descent directions with respect to the reference function  $F_w$  is convergent when adaptive step sizing is used. Thus,

a subroutine for step size determination should be embedded in the routine given in this chapter. Before concluding this section we will describe two step sizing routines which will yield a converging sequence of points.

(i) A line search subroutine which finds a point in the descent direction  $-d$ , which has a reduced  $F_w$  value. In formal terms,  $y^{k+1} = y^k - \alpha d$ ,  $\alpha > 0$  such that  $F_w(y^{k+1}) < F_w(y^k)$ . In this manner (following the steepest descent idea) one would obtain a monotonically decreasing sequence of points. Naturally, one should consider the sufficiency conditions of global convergence [Fletcher 87], [Dennis & Schn. 83].

(ii) An adaptive step sizing subroutine which maintains a current fraction of  $\alpha_{\max}$  as the step size and which is able to update this fraction when the general descent behavior of  $F_w$  is interrupted. To detect this phenomenon (orbiting), the algorithm can store the successive values of  $F_w$  for several iterations (say 5). If descent has not taken place in between more than half of these successive steps, then the current fraction value should be reduced. For example, an adaptive algorithm which takes the initial step size as  $\frac{\alpha_{\max}}{2}$  can adjust the current step size by multiplying with  $1/2$  when necessary. Here  $1/2$  may be replaced with any other suitable value between 0 and 1. Step sizes may also be determined by dividing  $\alpha_{\max}$  by any value greater than 1. Such a procedure acts as an overall reduction algorithm for  $F_w$ . This is because for all nonfeasible points  $F_w(y^k) > 0$ , descent conditions always exist, and upon decreasing the step size one will ultimately obtain (after  $N$  finite iterations) a new point for which  $F_w(y^{k+N}) < F_w(y^k)$ . Similarly, one should also check for the sufficiency conditions which guarantee the convergence of descent algorithms. The idea behind using an overall reduction procedure rather than a monotonic algorithm is to avoid the slow convergence rates observed in steepest descent type algorithms when the elliptic eccentricity is dominant.

Verification of convergence with the use of a subroutine as described in (i) is quite similar to that of a steepest descent routine. The claim of convergence for a subroutine of type (ii) can be verified in the following way. Suppose that we are at iteration  $k$ . After  $N_1$  iterations we should have  $F_w(y^k) > F_w(y^{k+N_1})$ . Similarly, after  $N_2$  iterations we should have  $F_w(y^k) > F_w(y^{k+N_1}) > F_w(y^{k+N_1+N_2})$  (see Figure 4.12). Provided that these incremental reductions satisfy the sufficiency conditions for subgradient descent algorithms, after a sequence of these states we should reach a point  $\bar{y}$  such that  $F_w(\bar{y}) = 0$ . At this point, we are either at the

interior of  $\Gamma$  or on the boundary of it. If  $\bar{y}$  is not interior, then we can move to an interior point in a single step, as described below.

In our discussions we have considered the problem to be almost solved, whenever we have  $F_w(\bar{y}) = 0$ , for some  $\bar{y}$ . At such an instant, either  $\forall i f_i(\bar{y}) < 0$  and thus  $\bar{y}$  is interior, or  $\forall i f_i(\bar{y}) \leq 0$ ,  $\exists i f_i(\bar{y}) = 0$ , and in this case  $\bar{y}$  is not an interior point of  $\Gamma$ . If this is the case, we can easily determine an interior point by a single step of the form  $y^+ = \bar{y} - \alpha d$  where  $0 < \alpha < \epsilon$ . By definition,  $\epsilon$  is the radius of the full dimensional ball contained in the convex body.

## Chapter 5

# Applications

Convex optimization has many real life applications, and possibly many more which are unrecognized [Boyd & Vand. 97]. The linear feasibility problem by itself, has many important applications. Image reconstruction from projections and tomography are probably the most studied ones.

In medical imaging and computerized tomography the so called algebraic reconstruction technique (ART) has been popular since the early seventies. This approach may be regarded to be equivalent to a discrete version of the inverse Radon transformation. From the computational point of view, one has to solve the following equation for  $f$ :

$$g = Hf + e \quad (5.1)$$

where  $e$  is an error term which is assumed to satisfy  $|e| < \epsilon$ . Equation (5.1) can be reduced to a feasibility problem of the form  $-\epsilon \leq g - Hf \leq \epsilon$ , which takes the generic form:

$$A\xi \leq b \quad (5.2)$$

for which a feasible solution  $\xi$  yields the digital image representing the object, subject to the specified error [Herman 80], [Natterer 85]. (To avoid confusion with the coordinate variable  $x$ , in this chapter the unknown vector will be denoted by  $\xi$ .) For this type of discrete image reconstruction applications the  $m \times n$  matrix  $A$  is a huge and sparse matrix without any definite structure. Similar formulations have also been devised in related fields such as magnetic resonance imaging [Liang & Laut. 91], ultrasound imaging [Rohl. *et al.* 97] and spectroscopic tomography [Sal.

& Sheh. 90], but the stated matrix properties are not necessarily valid in all of these cases. It has to be stated that the linear feasibility approach to tomography and similar problems are not competitive with sophisticated approaches such as filtered back-projection or Fourier transform methods [Censor & Herman 87], [Atalar 97]. It is the objective of this chapter to demonstrate image processing problems which cannot be solved efficiently by such techniques, but can be handled with the linear feasibility approach.

Although the above way of posing an inverse problem as a feasibility problem is most commonly mentioned in the context of tomography, we note that it is in fact much more general. Assume that we are able to measure indirectly a vector variable  $f$ , by some procedure  $Hf$  with some error. The system  $Hf = g$  may be overdetermined and—due to noise, errors in measurement, and discretization—possibly inconsistent. So one has to allow for a certain error tolerance to determine  $f$ . Thus writing as in [Censor & Herman 87]:

$$- \epsilon \leq g - Hf \leq \epsilon \quad (5.3)$$

with a sufficient error vector  $\epsilon$ , one can find a feasible solution  $f$  to this system.

Direct projection algorithms for solving the interval feasibility problem given in (5.3) which arises in tomography applications, have been devised. These algorithms have similar convergence properties. Several examples are given in [Censor & Zenios 97].

In this chapter we will discuss two image restoration problems which can be formulated as  $A\xi \leq b$ . Both problems deal with recovering blurred images in photographic films. In the first problem the image is blurred and distorted severely due to effects such as object or camera motion during the photographic exposure. In the second problem, the image is blurred by the so called point spread function (referred as PSF) arising from effects such as misfocus of the photographic device, atmospheric turbulence, etc. Additional effects which cause further distortions of the image, are also included in these models.

## 5.1 Image Recovery Formulated as a Linear Feasibility Problem

In this chapter, we show that two rather general classes of image recovery problems can be efficiently solved by formulating them as a linear feasibility problem. When discretized, the class of problems which we deal with is of the general linear form:

$$g = Hf \quad (5.4)$$

where  $g$  is the vector representing the observed image and  $f$  is the vector representing the original image to be recovered.  $H$  represents the system which distorts the image. If the images have  $N \times N$  pixels, then  $f$  and  $g$  are column vectors of size  $N^2$  obtained by stacking the columns of the image matrix on top of each other [Jain 89]. Thus  $H$  is an  $N^2 \times N^2$  matrix. (Generalization to rectangular images is straightforward.) For the  $160 \times 160$  images used in our examples, the matrix  $H$  has a size of  $25,600 \times 25,600$ . Normally, matrices of this and larger sizes cannot be handled unless they are sparse. The class of problems we deal with lead to such large, sparse matrices representing distortions and blurs which are:

1. *Nonseparable*. The two dimensions are coupled and the problem cannot be reduced to two one-dimensional problems.
2. *Anisotropic*. The distortion is different along different directions.
3. *Space variant*. The distortion is different for different parts of the image; it is not space invariant.
4. *Nonlocal*. The value of the distorted image at a certain point may depend on values of the original image at distant points; the distortion is of a global nature.

If the system is separable or isotropic or space invariant, then it may be possible to simplify the problem by exploiting the special structure of  $H$ . (For instance, see image processing texts such as [Jain 89], [Lim 90], [Pratt 91]. Also see [Fish *et al.* 96] for a discussion of the difficulties involved in the general case and an approach appropriate when the distortion is local.) In this chapter we assume that the matrix  $H$  does not exhibit any special structure which would allow some

reduction or decomposition techniques to be used. Furthermore, due to the large size of  $H$ , direct methods such as Wiener filtering or those based on singular value decomposition cannot be applied (they would require too much storage and time). Despite these difficulties, the approach taken in this chapter allows such problems to be solved efficiently.

Typically there will be an amount of measurement error or noise associated with the observation, which may lead to an inconsistent system of equations. Denoting the noisy observation by  $g'$ , we allow for such errors by writing:

$$|(g' - Hf)_i| \leq \epsilon \quad i = 1, \dots, N^2 \quad (5.5)$$

where  $(g' - Hf)_i$  is the  $i^{\text{th}}$  component of  $g' - Hf$  and  $\epsilon$  is a suitable error tolerance parameter, which we will quote as a percentage of the mean value of  $g$ .

Equation (5.5) can be converted into a feasibility problem of the form  $A\xi \leq b$  by setting

$$A = \begin{bmatrix} H \\ -H \end{bmatrix}_{2N^2 \times N^2} \quad b = \begin{bmatrix} \mathcal{E} + g' \\ \mathcal{E} - g' \end{bmatrix}_{2N^2 \times 1} \quad \xi = f_{N^2 \times 1} \quad (5.6)$$

where  $\mathcal{E}$  is an  $N^2 \times 1$  vector of  $\epsilon$ 's.

As we have stated previously, although the solution of a linear inequality system of the form  $A\xi \leq b$  with conventional (Kaczmarz or Cimmino type) projection algorithms is in principle straightforward, convergence is often exceedingly slow for large problems. This has led to the development of the *block projections* or *surrogate constraint* algorithms [Aharoni & Censor 89], [Yang & Murty 92], which has been the focus of this research.

## 5.2 Pre-filtering and Smoothing

When the relative measurement errors are very small, solution of equation (5.5) may directly yield a good restoration of the original image  $f$ . However in general, since we are dealing with a severely *ill conditioned* problem [Rush. 87], [Herman 80], [Sal. & Sheh. 90], [Fleming 90], even small observation errors will tend to show up as very large restoration errors. It is thus necessary to introduce some kind of *regularization* [Rush. 87] to limit the negative effects of noise to what is fundamentally unavoidable. Physically, what happens is that the severe distortions we consider blur the finer



details of the original image and those image components which are depressed below the noise level become irretrievably lost. However, unless care is exercised, the restoration process may amplify this noise, resulting in a restored image which is much worse than the best possible.

There are many approaches to regularizing ill conditioned inverse problems, most of which involve the introduction of some kind of a priori knowledge regarding the original image. This knowledge may take the form of additional constraints or statistical information. We will assume that such information is not available apart from the condition of *piecewise smoothness*, which means that the image consists of smooth regions separated by sharp edges. This is the most we will allow ourselves since our interest is in generic real images, for which even this condition is only partly valid.

Our regularization procedure consists of two components. In the first, called pre-filtering, we try to reduce the noise in  $g'$  as much as possible. We multiply the Fourier transform of  $g'$  by  $(1 + S_n/S_g)^{-1}$ , where  $S_n$  is the power spectral density of the noise (simply equal to the variance for zero mean independent identically distributed noise), and  $S_g$  is the magnitude squared of the Fourier transform of  $g$ . When  $S_n \ll S_g$ , the noise is small so that the filter is simply equal to 1. When  $S_n \gg S_g$ , noise is dominant and the filter is close to 0. In between, the filter effects a smooth transition. Thus, the overall effect of this filter is to eliminate the components of the image for which noise is dominant while preserving others. Of course, in reality we do not have access to  $g$ , but only  $g'$ . Thus we have used  $S_{g'} - S_n$  as a surrogate for  $S_g$  (setting it equal to zero for those frequencies where it goes negative). This is based on the assumption that the noise is not correlated with  $g$  so that  $S_{g'} = S_g + S_n$ . Filtering procedures of this nature are commonly used in image processing [Jain 89], [Lim 90], [Pratt 91].

The second component of our regularization procedure, smoothing, is applied after the feasibility problem is solved and a tentative (noisy) restoration  $f_{\text{tent}}$  is obtained. We introduce the following edge preserving nearest neighbor smoothing, inspired by edge preserving penalty functions [Delaney & Bresler 98]:

$$f_{\text{tent}} \rightarrow f_{\text{tent}} + \beta \Delta f_{\text{tent}} z(\Delta f_{\text{tent}}/\gamma) \quad (5.7)$$

where  $0 < \beta \leq 0.5$ ,  $\gamma > 0$ ,  $z(\zeta) = \arctan(\zeta)/\zeta$ , and  $\Delta f_{\text{tent}}$  is the difference between the average of the four nearest neighbors of a given image pixel, and the pixel itself.

The usual choice for  $\beta$  is 0.5 and the usual choice for  $\gamma$  is comparable or somewhat larger than the level of noise which we are trying to smooth out but sufficiently smaller than the edges or other features we are trying to protect. (Naturally, these two requirements sometimes conflict for the generic images we are considering, so that a compromise is necessary.) When  $\Delta f_{\text{tent}}/\gamma \ll 1$ ,  $z(\Delta f_{\text{tent}}/\gamma) \approx 1$  and the procedure becomes equivalent to taking the average of a given image pixel with the average of its neighbors. When  $\Delta f_{\text{tent}}/\gamma \gg 1$ ,  $z(\Delta f/\gamma)$  is small and the procedure results in little change. A third parameter of the smoothing process is the number of times ( $n$ ) equation (5.7) is consecutively applied. The overall effect is to smooth out the noise while limiting the damage to the edges.

### 5.3 Restoration of Space Variant Global Blurs Caused by Severe Camera Movements and Coordinate Distortions

Consider an original image  $f(q)$  which undergoes a general time varying, nonlinear coordinate distortion represented by  $q = \rho(r, t) = [u(r, t), v(r, t)]$  so that at time  $t$  the image is observed as  $f(\rho(r, t)) = f(u(r, t), v(r, t))$ . Here  $r$  stands for the position vector  $(x, y)$ . Now assume that we expose a photographic film to this time varying image for a duration of  $T$  seconds. The image  $g(r)$  recorded on the film is given by:

$$g(r) = K \int_{t=0}^T f(\rho(r, t)) dt \quad (5.8)$$

where  $K$  is a constant. (This equation may also have other physical interpretations than the one mentioned here. One such interpretation, the image reconstruction from projections problem, will be mentioned in Chapter 6.) To appreciate the generality of this observation model, we consider some of its special cases:

1. *Translational motion.*  $\rho(r, t) = r - r_c(t)$  where  $r_c(t) = [x_c(t), y_c(t)]$  is a given function representing the motion of the original image or object as a function of time. Arbitrary two-dimensional movement with arbitrary acceleration and higher order derivatives are allowed. (Although we speak of the movement of the object, it is clear that this also covers movement of the film or the camera.)

2. *Isotropic scaling.*  $\rho(r, t) = r/m(t)$  where  $m(t)$  is an arbitrary scaling function of time. More generally, different scaling functions can be assumed in the two dimensions (anisotropic or elliptic scaling). By properly choosing and interpreting  $m(t)$ , it is possible to model the movement of the object towards or away from the camera.
3. *Rotation.*  $\rho(r, t) = R_{\phi(t)}r$  where  $R_{\phi(t)}$  is the  $2 \times 2$  rotation matrix  $[\cos \phi(t), \sin \phi(t); -\sin \phi(t), \cos \phi(t)]$ . Here  $\phi(t)$  is an arbitrary function of time representing the angle of rotation. More generally, the  $2 \times 2$  matrix may take the form of an arbitrary nonorthogonal (parallelogram type) distortion represented by the matrix  $[a(t), b(t); c(t), d(t)]$ , which can be used to model moderate out of plane rotations of the object.

Other special cases and their combinations may also be considered.

Since equation (5.8) represents a linear relation between  $g$  and  $f$ , it is possible to write it in the form:

$$g(r) = \int_{r'} H(r, r') f(r') dr' \quad (5.9)$$

To find  $H(r, r')$ , we use  $f(\rho(r, t)) = \int_{r'} f(r') \delta(r' - \rho(r, t)) dr'$ , in equation (5.8) to obtain:

$$g(r) = \int_{r'} \left[ K \int_{t=0}^T \delta(r' - \rho(r, t)) dt \right] f(r') dr' \quad (5.10)$$

from which we conclude that

$$H(r, r') = K \int_{t=0}^T \delta(r' - \rho(r, t)) dt \quad (5.11)$$

Here  $\delta(r)$  is the so called delta function whose unit mass is concentrated at  $r = 0$  (see for example [Jain 89]).

The discrete counterpart of equation (5.9) is simply the general linear matrix equation  $g = Hf$ . Since equation (5.11) is a one-dimensional integral over time whereas the images are two-dimensional, the number of nonzero elements of  $H$  will typically be of the order of  $N^3$  (an average of  $N$  elements per row) out of  $N^2 \times N^2 = N^4$  elements. This sparsity property is what makes it possible to solve this class of problems despite the fact that the dimensions of  $H$  are very large. The large and sparse matrix  $H$  should be represented in sparse matrix format [Pissa. 84]. Since  $\rho(r, t)$  is arbitrary, the nonzero elements of  $H$  will in general be distributed quite irregularly and will not exhibit any special structure. We emphasize that unlike

many sparse matrix algorithms which rely on some special structure of the matrix, the approach described in Chapter 3 is particularly suited and effective in handling matrices whose nonzeros are arbitrarily distributed.

As an example we will consider restoration of an image distorted by the combined effects of translational and rotational motion, and elliptic scaling:

$$u(x, y, t) = \frac{\cos \phi(t) (x - x_c(t)) + \sin \phi(t) (y - y_c(t))}{m_u(t)} \quad (5.12)$$

$$v(x, y, t) = \frac{-\sin \phi(t) (x - x_c(t)) + \cos \phi(t) (y - y_c(t))}{m_v(t)} \quad (5.13)$$

where  $u(x, y, t)$  and  $v(x, y, t)$  are components of  $\rho(r, t)$ . We emphasize that there is nothing special about this particular example; the present approach can equally efficiently handle arbitrary nonlinear  $\rho(r, t)$ , which may even be discontinuous and fragmented. We will consider two cases. The functions  $x_c(t)$  and  $y_c(t)$  which define the translational motion are given in Figure 5.1 along with the trajectory itself. Notice that the motion is quite irregular and involves not only acceleration but also

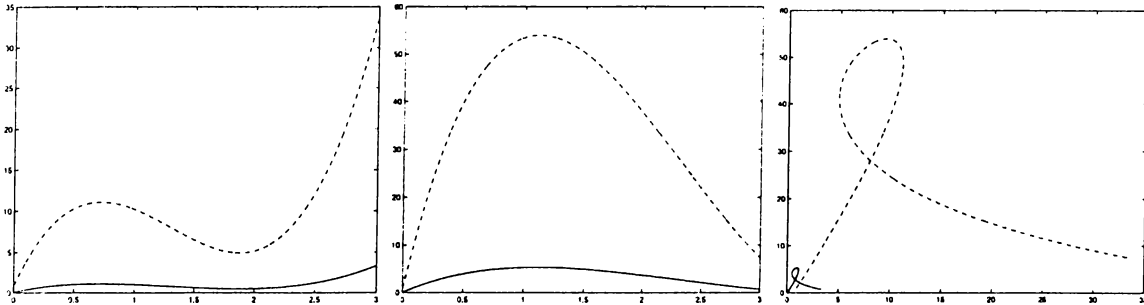


Figure 5.1: (a)  $x_c(t)$ . (b)  $y_c(t)$ . (c) The complete trajectory from  $t = 0$  to  $t = T = 3$ .  $x$  and  $y$  are measured in number of image pixels. (Case 1: solid line, case 2: dashed line.)

higher order derivatives. In case 1, the extent of the motion is small in comparison to the size of the images, whereas in case 2 it is comparable to the size of the images. The functions  $\phi(t)$ ,  $m_u(t)$  and  $m_v(t)$  are illustrated in Figure 5.2. In case 1, the object undergoes a moderate rotation of 5 degrees, whereas in case 2 it undergoes a 180 degrees rotation and ends upside down. Thus while case 1 represents a moderate local blur, in case 2 the image undergoes a severe blurring of a global nature. The original and the distorted images are shown in Figure 5.3. The restored images for the two cases are illustrated in Figure 5.4(a) and 5.5(a) when there is no (or negligible) measurement error. A relative tolerance of  $10^{-4}$  has been used.

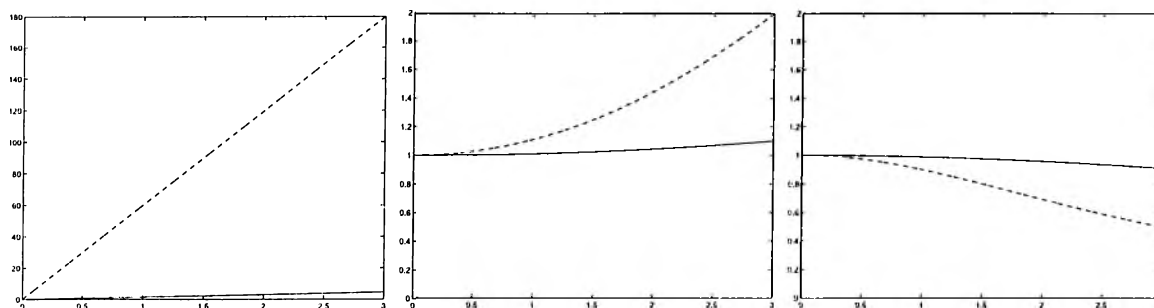


Figure 5.2: (a)  $\phi(t)$ . (b)  $m_u(t)$ . (c)  $m_v(t)$ . (Case 1: solid line, case 2: dashed line.)



Figure 5.3: (a) Original image. (b) Distorted image (case 1). (c) Distorted image (case 2).

For the examples where random noise is also included, the measurement errors were simulated by adding identically independent distributed zero mean normal random variables to  $g$  such that  $\epsilon = 2$  standard deviations. After pre-filtering the observation, the recovery problem can be formulated as a feasibility problem of the form  $A\xi \leq b$  as discussed in Section 5.1, and the feasibility problem can be solved by using the class of algorithms discussed in Chapter 3. Finally, smoothing is employed to obtain the recovered images shown in Figure 5.4(b,c) and 5.5(b,c). Table 5.1 shows the resulting mean restoration errors as a percentage of the mean image intensity.

In case 1, the distortion is moderate so that good restorations are possible even when the measurement error is relatively large. Although the distortion in case 2 is very severe, blurring the original beyond any possible recognition, quite good restorations are obtained when the noise is small. However, the restorations become less satisfactory as the measurement error is increased. This illustrates the fundamental tradeoff between the amounts of noise and blur which can be





Figure 5.4: Recovered Images for case 1: (a) Without measurement error. (b) With 0.5 % relative measurement error. (c) With 5 % relative measurement error.



Figure 5.5: Recovered Images for case 2: (a) Without measurement error. (b) With 0.5 % relative measurement error. (c) With 5 % relative measurement error.

simultaneously tolerated in ill conditioned problems. When the noise and blur are both large, a significant amount of information becomes lost and a faithful restoration is not possible.

Further improvements would be possible by introducing additional a priori knowledge in the form of additional constraints or statistical information, which we have assumed are not available. The condition of piecewise smoothness we have employed is only approximately valid for the generic images we have considered. This condition is much more effective when the original images consist of relatively large regions separated by sharp boundaries, and when the intensity within each region is constant or nearly constant [Delaney & Bresler 98].

	No error	0.5 %	5 %
Case 1	0.6 % (0,.-)	1.9 % (1, 0.5, 5 % )	5.3 % (2, 0.5, 10 % )
Case 2	3.2 % (0,.-)	5.6 % (4, 0.5, 10 % )	12.9 % (12, 0.5, 30 % )

Table 5.1: Restoration errors for case 1 (moderate blur) and case 2 (severe blur) for two different levels of the measurement error, quoted as a percentage of the mean value of  $g$ . The numbers in the parentheses are the values of the smoothing parameters  $n, \beta$  and  $\gamma$  used, where  $\gamma$  is expressed as a percentage of the mean image intensity.

#### 5.4 Image Recovery in the Presence of Severely Space Variant Geometric Distortions and Point Spread Functions

In this section we consider another very general image restoration model. We begin by introducing a basic point spread function (PSF)  $h(r)$ . It will be assumed that most of the mass of  $h$  is contained in a region whose area is small with respect to the images.  $h$  is otherwise totally arbitrary; it need not be localized around  $(0, 0)$  or any other point, and its mass may be divided among multiple component regions separated by arbitrary distances. We consider the general class of kernels  $H$  defined as

$$H(r, r') = h[u_{r'}(r), v_{r'}(r)] \quad (5.14)$$

where  $u$  and  $v$  are arbitrary nonlinear functions of  $r$  and  $r'$ , which introduce a  $(x', y')$ -dependent coordinate distortion (CD) into  $h(x, y)$ . (Special cases include space varying translations, scalings, rotations, and more generally, affine distortions.) Thus, equation (5.14) represents the combined effects of an arbitrary global CD and an arbitrary PSF. This form is capable of modeling a very large class of problems where the distortions and blurs may arise from diverse effects such as diffraction, misfocus and misalignment, coordinate transformations, different kinds of aberrations, design errors, atmospheric turbulence or other environmental factors, or their combinations.

We now reconsider the PSF  $h$ , but this time explicitly write it in the parametric form  $h(r; p_1, p_2, \dots, p_m)$ , where the parameters  $p_i$  are functions of  $r'$ . For instance,

$h$  may take the quadratic exponential form

$$h(x, y; A, B, C, \mu_1, \mu_2) = N e^{-[A(x-\mu_1)^2 + B(x-\mu_1)(y-\mu_2) + C(y-\mu_2)^2]} \quad (5.15)$$

where  $N$  normalizes the mass of  $h$  to unity. Now, we take

$$H(r, r') = h(r - r'; p_1, p_2, \dots, p_m) \quad (5.16)$$

Had the  $p_i$  been constants, this would reduce to the space invariant case. We do not even assume that the  $p_i$  are *slowly varying*; so that the present approach is able to handle cases of extreme or abrupt space variance as well. (Many treatments of space variant problems make a slowly varying assumption so that the problem can be approximately reduced to many smaller locally space invariant problems.)

Although our approach can handle both equation (5.14) and equation (5.16), the latter is probably more application oriented since it may be more convenient to model systems in terms of their deviations from space invariance. At the same time, this form retains full generality, since any  $H$  can be obtained by appropriately choosing  $h$  and  $p_i$ .

As an example, we use the basic PSF given in equation (5.15) and choose  $A$ ,  $B$ ,  $C$  such that the  $1/e$  contour is an ellipse centered at  $(\mu_1, \mu_2)$  whose first normal axis (along which it has radius  $a_1$ ) makes an angle  $\theta$  with the  $x$  axis and whose second normal axis (along which it has radius  $a_2$ ) makes an angle  $\theta$  with the  $y$  axis. The nature of the resulting blur is much easier to visualize in terms of the parameters  $a_1, a_2, \theta$  characterizing the contour of equation (5.15), as opposed to  $A, B, C$ . The formulas relating  $A, B, C$  to  $a_1, a_2, \theta$  are:

$$A = \left( \frac{1}{a_1^2} - \frac{1}{a_2^2} \right) \cos^2 \theta + \frac{1}{a_2^2} \quad (5.17)$$

$$B = 2 \left( \frac{1}{a_1^2} - \frac{1}{a_2^2} \right) \sin \theta \cos \theta \quad (5.18)$$

$$C = - \left( \frac{1}{a_1^2} - \frac{1}{a_2^2} \right) \cos^2 \theta + \frac{1}{a_1^2} \quad (5.19)$$

Note that  $h$  will now take the form  $h(r; a_1, a_2, \theta, \mu_1, \mu_2)$ . Now, it only remains to specify  $\mu_1, \mu_2, \theta, a_1, a_2$  as functions of  $r'$ . As an example, we consider

$$\mu_1 = - \cos \phi' K(0.5 + \sin^2 \phi') \rho'$$

$$\mu_2 = - \sin \phi' K(0.5 + \sin^2 \phi') \rho'$$



No error	0.2 %	1 %	5 %
2.0 % (0,-,-)	3.8 % (1, 0.5, 3 %)	5.2 % (3, 0.5, 5 %)	8.2 % (8, 0.5, 15 %)

Table 5.2: Restoration errors for three different levels of the measurement error, quoted as a percentage of the mean value of  $g$ . The numbers in the parentheses are the values of  $n$ ,  $\beta$  and  $\gamma$  used, where  $\gamma$  is expressed as a percentage of the mean image intensity.

$$\begin{aligned}
\theta &= \phi' \\
a_1 &= a_0 + k_1(1 + \sin^2 \phi')\rho'^2 \\
a_2 &= a_0 + k_2(1 + \cos^2 \phi')\rho'^2
\end{aligned} \tag{5.20}$$

where  $(\rho', \phi')$  is the polar representation of  $r' = (x', y')$  and  $K$ ,  $a_0$ ,  $k_1$ ,  $k_2$  are positive constants. All distances are measured in number of image pixels. The dependence of  $(\mu_1, \mu_2)$  on  $r'$  implies a so called barrel type distortion. The dependence of  $a_1$ ,  $a_2$  on  $\rho'$  implies an isotropic blur near the center of the field, which increases as we go towards the edges at a rate which is angle dependent, and resulting in an anisotropic blur.  $\theta = \phi'$  implies that, in our example, the axis along which the radius of the ellipse is  $a_1$  is always pointing in the radial direction. Overall, the distortion and blurring is of a highly irregular nature. In our example we take  $K = 8^{-1}$ ,  $a_0 = 6$ ,  $k_1 = 7200^{-1}$ ,  $k_2 = 9600^{-1}$ . The original and degraded images are shown in Figure 5.6(a,b). We emphasize that there is nothing special about this particular example; the present approach can effectively handle arbitrary nonlinear  $u$  and  $v$  in equation (5.14) or  $p_i$  in equation (5.16), which may even be discontinuous and fragmented.

The recovered image corresponding to zero measurement error is shown in Figure 5.6(c). A relative tolerance of  $10^{-4}$  has been used. Figures 5.7(a,b,c) show the recovered images for different levels of the measurement error, which was simulated by adding independent identically distributed zero mean normal random variables to  $g$  such that  $\epsilon = 2$  standard deviations. In this section, pre-filtering has not been applied since the blur which we have considered is not as severe as in the previous section and the use of pre-filtering did not offer any noticeable improvement. Smoothing was applied to obtain the results shown in Figures 5.7. Table 5.2 shows the resulting mean restoration errors as a percentage of the mean image intensity and the smoothing parameters used in equation (5.7). The fact that

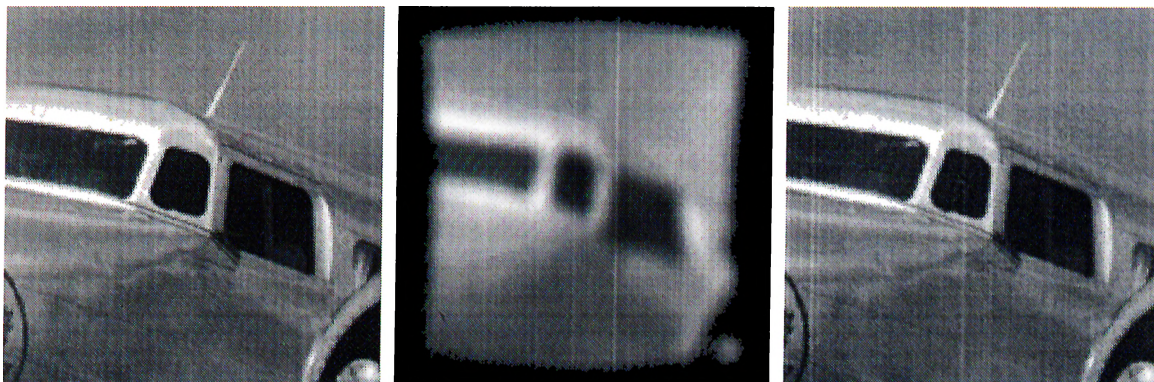


Figure 5.6: (a) Original image. (b) Degraded image. (c) Recovered image without measurement error.



Figure 5.7: Recovered images when the measurement error is (a) 0.2 %; (b) 1 %; (c) 5 %; quoted as a percentage of the mean value of  $g$ .

Figure 5.7(c) is relatively less satisfactory again illustrates the fundamental tradeoff between the amounts of noise and blur which can be simultaneously tolerated in ill conditioned problems. When the noise and blur are both large, a significant amount of information becomes lost and a faithful restoration is not possible, unless additional a priori knowledge is available.

## 5.5 Notes on Implementation of the Block Projections Algorithm in Image Restoration Applications

Block-iterative methods have been used in several image processing applications [Censor 88], [Piestun *et al.* 96]. In this study we have presented two novel application classes. In this section we will summarize our experience during the implementation

of the block projections approach.

In actual implementation, we have considered the interval feasibility problem  $g - \varepsilon \leq Hf \leq g + \varepsilon$ , following the idea given in [Censor & Zenios 97] instead of  $A\xi \leq b$ , since  $A$  is composed of  $H$  and  $-H$  and has twice as many rows as  $H$ . In this way, we saved both from memory storage and major iteration run time.

As stated previously, most image restoration and reconstruction problems are highly ill conditioned and as a consequence the iterative algorithms converge quite slowly. During implementation it is necessary to stop the algorithm when further progress falls below a certain level (infeasible termination). Due to the ill conditioned nature of the matrix, in the presence of noise one should consider some kind of regularization to limit the amplification of noise in the final result. Conventional tools like singular value decomposition (see [Strang 88]) are out of the question for such large matrices.

For the examples given in Sections 5.3 and 5.4 we have only considered the successive block projections algorithm. In the meantime, simultaneous implementation (with the long steps discussed in Section 3.5) has been given in [Turna 98] with encouraging results, in terms of speedup and efficiency. It has to be stated that the algorithmic behavior depends highly on the parameters of the algorithm for the ill conditioned image processing applications considered in this chapter. This issue deserves further research and will be dealt with elsewhere.

In our current implementations, solutions are obtained in the order of hours (instead of minutes as for the random problems in Chapter 3) and ten thousands of iterations. Still, we can hope for significant improvements over these preliminary results through parametric experimentation and parallel computing.

The algorithmic behavior for inconsistent systems (when measurement errors are included) is quite interesting. It turns out that the algorithm always converges towards some approximate solution. Thus for the block projections approach we obtain results similar to those obtained by [De Pierro & Iusem 85] for Cimmino's method and [Hanke & Niet. 90] for Kaczmarz' method, when applied to infeasible problems.

## Chapter 6

# Conclusions and Prospects for the Future

In Chapter 3 we have presented a thorough experimental study of the block projections approach. Our results indicate that the long-step simultaneous block projections algorithm is quite competitive with the successive block projections algorithm. It should be recalled that the conventional short-step simultaneous block projections algorithm is quite poor in performance when compared to its sequential counterpart. So with the use of long steps, it is possible to benefit from distributed implementation on parallel processors.

In the random test problems solved in Chapter 3 we observed that the simultaneous block projections method of Yang and Murty (without any adjustment) is quite poor when compared to the successive block projections method. This is because the magnitudes of the steps obtained in the parallel algorithm remain quite small with respect to the accumulated sequential steps. In order to compensate for this deficiency, we used the longer step size given in (3.20) and obtained favorable results. We may conclude that the new step given by equation (3.20) (or more generally by (3.27)) should replace the conventional step (the convex combination of simultaneous projections) for all Cimmino based algorithms. The theoretical results given in this study are also valid for any simultaneous method for the convex feasibility problem which proceeds by projecting onto valid supersets.

Implementations employing the step size given in equation (3.20) have yielded many interesting results. It is observed that convergence is very sensitive to changes in the relaxation parameter  $\lambda_k$ . Best results are obtained when  $\lambda_k$  is chosen to have values close to 2.

Our main contribution in Chapter 4 is an algorithmic scheme which utilizes both cutting planes and subgradient movement directions. The motivating idea behind this approach is to obtain deep cuts to accelerate convergence, by utilizing the information contained in the subgradient directions. Although only theoretical results are presented, we find it worthwhile to proceed with experimental implementation in the near future.

In Chapter 4 it has been assumed that the underlying functions defining the convex set are differentiable. Despite its convenience, this assumption is not critical in establishing convergence. Thus we can conclude that the results are also valid for convex nondifferentiable functions where subgradients are provided by the separation oracle.

Further research will focus on implementation of routines with variable step sizing policies. The use of varying steps together with multiple test points is a fruitful research direction. Independent searches through many test points with the aid of parallel processors will conceptually and practically yield much more powerful algorithms, since the search area increases proportionally with the number of simultaneously implemented test points.

Although  $-d$  is not a descent direction for all individual  $f_i$ 's, it is so for the reference function  $F_w$ . What is nice about  $F_w$  is the fact that  $-d$  is the gradient (or a subgradient) for this function, at the test point  $\bar{y}$ . Further analysis of this function might possibly yield interesting results on the behavior of the algorithm. Another point worth noting is that the convex feasibility problem in Chapter 4 has been solved by applying a sort of steepest descent procedure on the weighted functional sum  $F_w$ . This suggests that the presented algorithm can also be used for multi-objective (convex) minimization problems.

In Chapter 5 we have shown that rather general classes of image blurring and distortion problems can be formulated as a linear feasibility problem of the form  $A\xi \leq b$ . The feasibility problem approach has been previously used for medical

image reconstruction (computerized tomography) [Herman 80]. It is not difficult to see that the tomography problem [Jain 89] is a special case of the general model introduced in equation (5.8):

$$g(s, \theta) = \int_t f(\rho(s, \theta, t)) dt = \int_t f(u(s, \theta, t), v(s, \theta, t)) dt \quad (6.1)$$

where

$$u(s, \theta, t) = s \cos \theta - t \sin \theta \quad (6.2)$$

$$v(s, \theta, t) = s \sin \theta + t \cos \theta \quad (6.3)$$

Here  $t$  is no longer time but the integration variable along the path of X-rays. We emphasize that, in Chapter 5 we allowed  $u(r, t)$  and  $v(r, t)$  to be arbitrary nonlinear functions which may even be discontinuous and fragmented.

The image processing applications we have addressed are in general nonseparable, anisotropic, global and space variant. These problems lead to large, sparse matrices (with the order of  $10^9$  elements for  $160 \times 160$  pixel images) whose nonzeros are irregularly distributed. The block projections algorithms employed are particularly suited for such situations. This match between these difficult problems and their effective solution is the main motivation for considering these applications.

Image restoration and reconstruction problems are often ill conditioned. There exists a fundamental tradeoff between the amounts of distortion and noise that can be simultaneously tolerated in an ill conditioned problem. The approach presented can be especially useful on that side of this tradeoff where the distortion is of a severe and very irregular nature, but the noise is low or moderate. A number of generalizations of the approach given in Chapter 5 can be readily implemented. If the imaging device is subjected to motion blur and also has an arbitrary space varying point spread function (PSF) of limited extent, then the combined effects of these two blurs can be incorporated into the matrix  $H$ . The only limitation arises from the fact that this will increase the number of nonzero elements of  $H$  (especially if the PSF is relatively broad), resulting in memory problems. What ultimately matters is the largest number of nonzero elements that the computer can handle. Another advantage of the approach is that it is possible to accommodate situations in which the observed data is incomplete or partial, or when additional a priori information about the original image exists. Thus we may conclude that our results may contribute to greater and more effective use of this type of algorithms for such image processing applications.

# Bibliography

- [Agmon 54] S. Agmon, The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6 (1954), pp. 382–392.
- [Aharoni & Censor 89] R. Aharoni and Y. Censor, Block-iterative projection methods for parallel computation of solutions to convex feasibility problems. *Linear Algebra and its Applications*, 120 (1989), pp. 165–175.
- [Altman & Kiwiel 96] A. Altman and K. C. Kiwiel, A note on some analytic center cutting plane methods for convex feasibility and minimization problems. *Computational Optimization and Applications*, 5 (1996), pp. 175–180.
- [Atalar 97] E. Atalar, Personal Communication, December 1997.
- [Atkinson & Vaidya 95] D. S. Atkinson and P. M. Vaidya, A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69 (1995), pp. 1–43.
- [Bahn *et al.* 94] O. Bahn, O. du Merle, J.-L. Goffin, and J.-P. Vial, A cutting plane method from analytic centers for stochastic programming. Technical Report, GERAD, September 1994.
- [Bauschke & Borwein 96] H. H. Bauschke and J. M. Borwein, On projection algorithms for solving convex feasibility problems. *SIAM Review*, 38 (1996), pp. 367–426.
- [Boyd & Vand. 97] S. Boyd and L. Vandenberghe, *Convex Optimization* Lecture Notes, Stanford University, Stanford, California, 1997.
- [Censor 88] Y. Censor, Parallel application of block-iterative methods in medical imaging and radiation therapy. *Mathematical Programming*, 42 (1988), pp. 307–325.

- [Censor & Herman 87] Y. Censor and G. T. Herman, On some optimization techniques in image reconstruction from projections. *Applied Numerical Mathematics*, 3 (1987), pp. 365–391.
- [Censor & Lent 82] Y. Censor and A. Lent, Cyclic subgradient projections. *Mathematical Programming*, 24 (1982), pp. 233–235.
- [Censor & Zenios 97] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms and Applications*. Oxford University Press, New York, 1997.
- [Chvátal 83] V. Chvátal, *Linear Programming*. W. H. Freeman and Company, New York, 1983.
- [Combettes 93] P. L. Combettes, The foundations of set theoretic estimation. *Proceedings of the IEEE*, 81 (1993), pp. 182–208.
- [Crombez 91] G. Crombez, Image recovery by convex combinations of projections. *Journal of Mathematical Analysis and Applications*, 155 (1991), pp. 413–419.
- [Dantzig & Eaves 73] G. B. Dantzig and B. C. Eaves, Fourier-Motzkin elimination and its dual. *Journal of Combinatorial Theory A*, 14 (1973), pp. 288–297.
- [De Pierro & Iusem 85] A. R. De Pierro and A. N. Iusem, A simultaneous projections method for linear inequalities. *Linear Algebra and its Applications*, 64 (1985), pp. 243–253.
- [Delaney & Bresler 98] A. H. Delaney and Y. Bresler, Globally convergent edge-preserving regularized reconstruction: An application to limited-angle tomography. *IEEE Transactions on Image Processing*, 7 (1998), pp. 204–221.
- [den Hertog *et al.* 90] D. den Hertog, C. Roos, and T. Terlaky, A potential reduction variant of Renegar’s short-step path-following method for linear programming. Technical Report 90-14, Delft University of Technology, Delft, The Netherlands, January 1990.
- [Dennis & Schn. 83] J. E. Dennis Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, New Jersey, 1983.
- [Dos Santos 87] L. T. Dos Santos, A parallel subgradient method for the convex feasibility problem. *Journal of Computational and Applied Mathematics*, 18 (1987), pp. 307–320.



- [Fish *et al.* 96] D. A. Fish, J. Grochmalicki, and E. R. Pike, Scanning singular-value decomposition method for restoration of images with space-variant blur. *Journal of Optical Society of America A*, 13 (1996), pp. 464–469.
- [Fleming 90] H. E. Fleming, Equivalence of regularization and truncated iteration in the solution of ill-posed image reconstruction problems. *Linear Algebra and its Applications*, 130 (1990), pp. 133–150.
- [Fletcher 87] R. Fletcher, *Practical Methods of Optimization*, second edition. John Wiley & Sons. New York, 1987.
- [Gar.-Pal. 90] U. M. García-Palomares, A class of methods for solving large convex systems. *Operations Research Letters*, 9 (1990), pp. 181–187.
- [Gar.-Pal. 93] U. M. García-Palomares, Parallel Projected Aggregation Methods for Solving the Convex Feasibility Problem. *SIAM Journal on Optimization*, 3–4 (1993), pp. 882–900.
- [Gar.-Pal. & Gon.-Cas. 96] U. M. García-Palomares and F. J. González Castaño, Acceleration technique for solving convex (linear) systems via projection methods. Technical Report OP960614, Universidade de Vigo, Escola Técnica Superior de Enxeñeiros de Telecomunicación, Lagoas Marcosende 36200 Vigo, España, 1996.
- [Geist *et al.* 94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*. The MIT Press, Cambridge, Massachusetts, 1994.
- [Goffin 80] J.-L. Goffin, The relaxation method for solving systems of linear inequalities. *Mathematics of Operations Research*, 5 (1990), pp. 388–414.
- [Gof. *et al.* 93a] J.-L. Goffin, Z.-Q. Luo, and Y. Ye, On the complexity of a column generation algorithm for convex or quasiconvex feasibility problems. Technical Report 93-17, GERAD, June 1993.
- [Gof. *et al.* 93b] J.-L. Goffin, Z.-Q. Luo, and Y. Ye, Further complexity analysis of a primal-dual column generation algorithm for convex or quasiconvex feasibility problems. Technical Report, GERAD, November 1993.

- [Gonzaga 92] C. C. Gonzaga, Path-following methods for linear programming. *SIAM Review*, 34 (1992), pp. 167–224.
- [Gritz. & Klee 93a] P. Gritzmann and V. Klee. Computational complexity of inner and outer  $j$ -radii of polytopes in finite-dimensional normed spaces. *Mathematical Programming*, 59 (1993), pp. 163–213.
- [Gritz. & Klee 93b] P. Gritzmann and V. Klee. Mathematical programming and convex geometry. In *Handbook of Convex Geometry*, Elsevier Science Publishers, 1993, pp. 627–674.
- [Grötschel *et al.* 88] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, 1988.
- [Hanke & Niet. 90] M. Hanke and W. Niethammer, On the acceleration of Kaczmarz's method for inconsistent linear systems. *Linear Algebra and its Applications*, 130 (1990), pp. 83–98.
- [Hir.-Urr. & Lemar. 93] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*. Springer-Verlag, Berlin, 1993.
- [Herman 80] G. T. Herman, *Image Reconstruction from Projections*. Academic Press, New York, 1980.
- [Jain 89] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice-Hall, New Jersey, 1989.
- [Kaiser *et al.* 91] M. J. Kaiser, T. L. Morin, and T. B. Trafalis, Centers and invariant points of convex bodies. *DIMACS Series in Discrete Mathematics and Theoretical computer Science*, 4 (1991), pp. 367–385.
- [Khachiyan 79] L. G. Khachiyan, A polynomial algorithm for linear programming. *Doklady Akad. Nauk. USSR*, 244 (1979), pp. 1093–1096.
- [Khachiyan & Todd 90] L. G. Khachiyan and M. J. Todd, On the complexity of approximating the maximal inscribed ellipsoid for a polytope. Technical Report 893, School of Operations Research and Industrial Engineering, College of Engineering, Cornell University, Ithaca, New York, February 1990.
- [Kiwiel 85] K. C. Kiwiel, A descent method for nonsmooth convex multiobjective minimization. *Large Scale Systems*, 8 (1985), pp. 119–129.

- [Kiwiel 95] K. C. Kiwiel, Block-iterative surrogate projection methods for convex feasibility problems. *Linear Algebra and its Applications*, 215 (1995), pp. 225–260.
- [Kiwiel 96a] K. C. Kiwiel, The efficiency of subgradient projection methods for convex optimization, part I: general level methods. *SIAM Journal of Control and Optimization*, 34 (1996), pp. 660–676.
- [Kiwiel 96b] K. C. Kiwiel, The efficiency of subgradient projection methods for convex optimization, part II: implementations and extensions. *SIAM Journal of Control and Optimization*, 34 (1996), pp. 677–697.
- [Kojima *et al.* 93] M. Kojima, N. Megiddo, and S. Mizuno, A primal-dual infeasible-interior-point algorithm for linear programming. *Mathematical Programming*, 61 (1993), pp. 263–280.
- [Koltracht *et al.* 90] I. Koltracht, P. Lancaster, and D. Smith, The structure of some matrices arising in tomography. *Linear Algebra and its Applications*, 130 (1990), pp. 193–218.
- [Liang. & Lauterbur 91] Z.-P. Liang and P. C. Lauterbur, A generalized series approach to MR spectroscopic imaging. *IEEE Transactions on Medical Imaging*, 10 (1991), pp. 132–137.
- [Lim 90] J. S. Lim, *Two-Dimensional Signal and Image Processing*. Prentice-Hall, New Jersey, 1990.
- [Luo & Sun 95] Z.-Q. Luo and J. Sun, An analytic center based column generation algorithm for convex quadratic feasibility problems. Technical Report, November 1995.
- [Luen. 84] D. G. Luenberger, *Linear and Nonlinear Programming*, second edition. Addison-Wesley, Reading, Massachusetts, 1984.
- [Mc Corm. 83] G. P. Mc Cormick, *Nonlinear Programming*. John Wiley & Sons, New York, 1983.
- [Mizuno *et al.* 90] S. Mizuno, M. J. Todd, and Y. Ye, On adaptive-step primal-dual interior-point algorithms for linear programming. Technical Report 944, School of Operations Research and Industrial Engineering, College of Engineering, Cornell University, Ithaca, New York, October 1990.

- [Mont. & Adler 89] R. D. C. Monteiro and I. Adler, Interior path following primal-dual algorithms: Part I: Linear programming. *Mathematical Programming*, 44 (1989), pp. 27-41.
- [Motzkin & Scho. 54] I. S. Motzkin and I. J. Schoenberg, The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6 (1954), pp. 393-404.
- [Natterer 86] F. Natterer. *The Mathematics of Computerized Tomography*. John Wiley & Sons, Stuttgart, 1986.
- [Nemirovsky & Yudin 83] A. S. Nemirovsky and D. B. Yudin, *Problem Complexity and Method Efficiency in Optimization*. John Wiley & Sons, Great Britain, 1983.
- [Oettli 72] W. Oettli, An iterative method, having linear rate of convergence, for solving a pair of dual linear programs. *Mathematical Programming*, 3 (1972), pp. 302-311.
- [Oko 92] S. O. Oko, Surrogate methods for linear inequalities, *Journal of Optimization Theory and Applications*, 72 (1992), pp. 247-271.
- [Özak. 95] H. Özaktaş, An introductory analysis of the cutting plane approach for the convex feasibility problem. Technical Report IEOR-9502, Department of Industrial Engineering, Bilkent University, Ankara, January 1995.
- [Özak. 96] H. Özaktaş, The rectangular cutting plane approach to the feasibility problem. Technical Report IEOR-9610, Department of Industrial Engineering, Bilkent University, Ankara, February 1996.
- [Özak. et al. 96] H. Özaktaş, M. Akgül, and M. Ç. Pınar, The parallel surrogate constraint approach to the linear feasibility problem. In *Lecture Notes in Computer Science*, vol. 1184, J. Waśniewski, J. Dongarra, K. Madsen, and D. Olesen, eds., Springer-Verlag, Berlin, 1996, pp. 565-574.
- [Özak. et al. 97a] H. Özaktaş, M. Ç. Pınar, and M. Akgül, An algorithm with long steps for the simultaneous block projections approach for the linear feasibility problem. Technical Report IEOR-9703, Department of Industrial Engineering, Bilkent University, Ankara, March 1997 (revised June 1998), submitted to *Journal of Information and Optimization Sciences*.

- [Özak. *et al.* 97b] H. Özaktaş, M. Ç. Pınar, and M. Akgül, An improved simultaneous block projections algorithm for the linear feasibility problem. In *Proceedings of the Twelfth International Symposium on Computer and Information Sciences*, S. Kuru, M. U. Çağlayan, and H. L. Akın, eds., published by Boğaziçi University, İstanbul, 1997, pp. 551–558.
- [Özak. *et al.* 98a] H. Özaktaş, M. Ç. Pınar, and M. Akgül, Restoration of space variant global blurs caused by severe camera movements and coordinate distortions. Technical Report IEOR-9818, Department of Industrial Engineering, Bilkent University, Ankara, June 1998, to appear in *Journal of Optics*.
- [Özak. *et al.* 98b] H. Özaktaş, M. Ç. Pınar, and M. Akgül, Image recovery in the presence of severely space variant geometric distortions and point spread functions. Technical Report IEOR-9819, Department of Industrial Engineering, Bilkent University, Ankara, July 1998.
- [Piestun *et al.* 96] R. Piestun, B. Spektor, and J. Shamir. Wave fields in three dimensions: analysis and synthesis. *Journal of Optical Society of America A*, 13 (1996), pp. 1837–1848.
- [Pissa. 84] S. Pissanetzky, *Sparse Matrix Technology*. Academic Press, London, 1984.
- [Pratt 91] W. K. Pratt, *Digital Image Processing*, second edition. John Wiley & Sons, New York. 1991.
- [Renegar 88] J. Renegar, A polynomial-time algorithm, based on Newton's method for linear programming. *Mathematical Programming*, 40 (1988), pp. 59–93.
- [Rohl. *et al.* 97] R. Rohling, A. Gee, and L. Berman, Three-dimensional spatial compounding of ultrasound images. *Medical Image Analysis*, 1 (1997), pp. 177–193.
- [Ross 83] S. M. Ross, *Stochastic Processes*. John Wiley & Sons, New York, 1983.
- [Rush. 87] C. K. Rushforth, Signal Restoration, Functional Analysis, and Fredholm Integral Equations of the First Kind. In *Image Recovery: Theory and Application*, H. Stark, editor, Academic Press, Orlando, Florida, 1987, pp. 1–27.

- [Sal. & Sheh. 90] M. Kh. Salakhov and N. K. Sheherbakova, A regularized algorithm for local emission reconstruction in spectroscopic tomography. *Linear Algebra and its Applications*, 130 (1990), pp. 219–229.
- [Schein. & Oliv. 92] S. Scheimberg and P. R. Oliveira, Descent algorithms for a class of convex nondifferentiable functions. *Journal of Optimization Theory and Applications*, 72 (1992), pp. 269–297.
- [Schr. 86] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, 1986.
- [Turna 98] E. Turna, *Parallel Algorithms for the Solution of Large and Sparse Inequality Systems on Distributed Memory Architectures*. M.S. Thesis, Department of Computer Science, Bilkent University, Ankara, August 1998.
- [Strang 88] G. Strang, *Linear Algebra and its Applications*. Harcourt-Brace-Jovanovich Publishers, San Diego, 1988.
- [Tarasov *et al.* 88] S. P. Tarasov, L. G. Khachiyan, and I. I. Erlikh, The method of inscribed ellipsoids. *Soviet Math. Doklady*, 37 (1988), pp. 226–230.
- [Todd & Ye 90] M. J. Todd and Y. Ye, A centered projective algorithm for linear programming. *Mathematics of Operations Research*, 15 (1990), pp. 508–529.
- [Vaidya 89] P. M. Vaidya, A new algorithm for minimizing convex functions over convex sets. Technical report, AT&T Bell Laboratories, Murray Hill, New Jersey, July 1989.
- [Yang & Murty 92] K. Yang and K. G. Murty, New iterative methods for linear inequalities. *Journal of Optimization Theory and Applications*, 72 (1992), pp. 163–185.
- [Ye 89] Y. Ye, A potential reduction algorithm allowing column generation. Technical Report, Department of Management Sciences, The University of Iowa, Iowa City, Iowa, July 1989.

# Vita

Hakan Özaktaş was born in Ankara on November 26, 1969. He graduated from T.E.D. Ankara College High School in 1986. He obtained the B.S. degree in Industrial Engineering from Bilkent University in 1990. He has been a graduate student in the same department since then. Throughout this period he has been employed as a research and teaching assistant and part-time instructor. He received the M.S. degree in 1992, with a thesis titled *Mathematical Models of Evolution*.